阿里云
日志服务

用户指南

文档版本：20181116

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止：** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告：** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明：** 您也可以通过按**Ctrl** + **A**选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

法律声明.....I
通用约定.....I
1 隐藏文件夹.....1
1.1 快速安装.....1
1.2 Syslog.....1
1.3 Syslog-采集参考.....5
1.4 实时分析.....11
2 准备工作.....12
2.1 准备流程.....12
2.2 操作Project.....14
2.3 操作Logstore.....18
2.4 操作Shard.....22
3 数据采集.....27
3.1 采集方式.....27
3.2 采集加速.....29
3.2.1 简介.....29
3.2.2 开启全球加速.....34
3.2.3 关闭全球加速.....39
4 Logtail采集.....41
4.1 简介.....41
4.1.1 Logtail简介.....41
4.1.2 Logtail采集原理.....45
4.1.3 Logtail配置和记录文件.....48
4.2 选择网络.....57
4.3 安装.....60
4.3.1 Linux.....60
4.3.2 Windows.....73
4.3.3 配置启动参数.....77
4.4 机器组.....81
4.4.1 简介.....81
4.4.2 创建IP地址机器组.....82
4.4.3 创建用户自定义标识机器组.....85
4.4.4 为非本账号ECS、自建IDC配置AliUid.....90
4.4.5 管理采集配置.....92
4.4.6 管理机器组.....94
4.5 文本日志.....98

4.5.1 采集文本日志.....98
4.5.2 配置解析.....108
4.5.3 配置时间格式.....110
4.5.4 导入历史日志文件.....112
4.5.5 生成主题.....115
4.6 自定义插件.....117
4.6.1 简介.....117
4.6.2 MySQL Binlog方式.....119
4.6.3 JDBC查询结果.....128
4.6.4 HTTP方式.....133
4.6.5 Syslog输入源.....137
4.6.6 Beats和Logstash输入源.....143
4.6.7 处理采集数据.....147
4.7 容器日志采集.....161
4.7.1 容器文本日志.....161
4.7.2 容器标准输出.....169
4.7.3 标准Docker日志采集流程.....179
4.7.4 Kubernetes日志采集流程.....184
4.7.5 Kubernetes-CRD配置日志采集.....192
4.8 相关限制说明.....201
5 云产品采集.....205
5.1 API网关访问日志.....205
5.2 MNS日志.....208
5.3 OSS访问日志.....214
5.4 DDoS高防日志.....229
5.4.1 简介.....229
5.4.2 采集步骤.....233
5.4.3 日志分析.....239
5.4.4 日志报表.....257
5.4.5 费用说明.....266
5.5 ActionTrail访问日志.....271
5.5.1 简介.....272
5.5.2 操作步骤.....276
5.6 NAS访问日志.....281
5.6.1 简介.....281
5.6.2 操作步骤.....285
5.7 态势感知日志.....288
5.8 DRDS SQL审计日志.....290
5.8.1 简介.....291
5.8.2 开启SQL日志审计.....294

5.8.3 日志字段 297
5.8.4 日志分析 298
5.8.5 日志报表 306
5.8.6 高级管理 311
5.8.7 为子账号授予DRDS SQL审计权限 314
6 其他采集方式 317
6.1 Web Tracking 317
6.2 Logstash 321
6.2.1 安装 321
6.2.2 配置 Logstash 为 Windows Service 322
6.2.3 创建 Logstash 采集配置 324
6.2.4 进阶功能 326
6.2.5 Logstash 错误处理 326
6.3 SDK采集 327
6.3.1 Producer Library 327
6.3.2 Log4j Appender 329
6.3.3 C Producer Library 329
6.4 常见日志格式 330
6.4.1 Apache 日志 330
6.4.2 Nginx 日志 335
6.4.3 Python日志 336
6.4.4 Log4j日志 341
6.4.5 Node.js日志 343
6.4.6 wordpress 日志 345
6.4.7 分隔符日志 346
6.4.8 ThinkPHP 日志 351
6.4.9 Logstash 收集 IIS 日志 352
6.4.10 Logstash 收集 CSV 日志 353
7 查询与分析 357
7.1 简介 357
7.2 实时分析简介 359
7.3 开启并配置索引 362
7.4 查询日志 368
7.5 索引数据类型 373
7.5.1 简介 374
7.5.2 文本类型 376
7.5.3 Json类型 378
7.5.4 数值类型 379
7.6 查询语法与功能 379
7.6.1 查询语法 379

7.6.2 LiveTail....385
7.6.3 上下文查询....390
7.6.4 快速查询....393
7.6.5 快速分析....395
7.6.6 其他功能....399
7.7 关联外部数据源....404
7.7.1 简介....404
7.7.2 关联MySQL数据源....406
7.7.3 关联OSS数据源....408
7.8 分析语法与函数....411
7.8.1 通用聚合函数....411
7.8.2 安全检测函数....412
7.8.3 Map映射函数....415
7.8.4 估算函数....417
7.8.5 数学统计函数....418
7.8.6 数学计算函数....419
7.8.7 字符串函数....421
7.8.8 日期和时间函数....422
7.8.9 URL函数....428
7.8.10 正则式函数....428
7.8.11 JSON函数....429
7.8.12 类型转换函数....430
7.8.13 IP地理函数....431
7.8.14 GROUP BY 语法....433
7.8.15 窗口函数....435
7.8.16 HAVING语法....437
7.8.17 ORDER BY语法....438
7.8.18 LIMIT语法....438
7.8.19 CASE WHEN和IF分支语法....439
7.8.20 嵌套子查询....440
7.8.21 数组....441
7.8.22 二进制字符串函数....443
7.8.23 位运算....444
7.8.24 同比和环比函数....444
7.8.25 比较函数和运算符....448
7.8.26 lambda函数....450
7.8.27 逻辑函数....453
7.8.28 列的别名....454
7.8.29 Logstore和RDS联合查询....455
7.8.30 空间几何函数....457

7.8.31 地理函数.....460
7.8.32 Join语法.....461
7.8.33 unnest语法.....462
7.9 机器学习语法与函数.....468
7.9.1 简介.....468
7.9.2 平滑函数.....471
7.9.3 多周期估计函数.....475
7.9.4 变点检测函数.....477
7.9.5 预测与异常检测函数.....480
7.9.6 序列分解函数.....485
7.9.7 时序聚类函数.....486
7.9.8 频繁模式统计函数.....492
7.9.9 差异模式统计函数.....493
7.10 分析进阶.....495
7.10.1 优秀分析案例.....495
7.10.2 优化查询.....496
7.10.3 查询分析全方位对比（ELK）.....497
7.10.4 查询方案（ELK/Hive）对比.....508
7.11 通过JDBC协议分析日志.....510
8 可视化分析.....514
8.1 分析图表.....514
8.1.1 图表说明.....514
8.1.2 仪表盘.....515
8.1.3 仪表盘过滤器.....522
8.1.4 下钻分析.....529
8.1.5 Markdown图表.....536
8.1.6 表格.....542
8.1.7 折线图.....545
8.1.8 柱状图.....548
8.1.9 条形图.....550
8.1.10 饼图.....551
8.1.11 单值图.....554
8.1.12 矩形树图.....560
8.1.13 地图.....561
8.1.14 面积图.....565
8.1.15 流图.....567
8.1.16 桑基图.....568
8.1.17 词云.....571
9 告警.....573
9.1 简介.....573

9.2 设置告警 575
9.3 通知方式 579
10 实时消费 586
10.1 实时消费简介 586
10.2 普通消费 587
10.3 消费组消费 588
10.3.1 消费组消费 588
10.3.2 消费组状态 594
10.3.3 消费组监控与报警 597
10.4 函数计算消费 599
10.4.1 开发指南 599
10.4.2 配置函数计算消费日志 603
10.5 Flink 消费 611
10.6 Storm消费 618
10.7 Spark Streaming消费 622
10.8 StreamCompute消费 622
10.9 CloudMonitor消费 623
10.10 ARMS消费 623
11 服务监控 624
11.1 监控日志服务 624
11.2 服务日志 625
11.2.1 简介 625
11.2.2 配置步骤 627
11.2.3 日志类型 633
11.3 云监控方式 640
11.3.1 云监控指标 640
11.3.2 设置云监控告警规则 643
12 数据投递 651
12.1 简介 651
12.2 投递日志到OSS 651
12.2.1 投递流程 651
12.2.2 CSV格式 658
12.2.3 RAM授权 661
12.3 投递日志到MaxCompute 663
12.3.1 通过DataWorks投递数据到MaxCompute 663
12.3.2 通过日志服务投递日志到MaxCompute 672
12.4 管理日志投递任务 681
13 访问控制 RAM 683
13.1 简介 683

13.2 授权RAM 用户.....685
13.3 RAM自定义授权场景.....686
13.4 授权服务角色.....691
13.5 授权用户角色.....693

1 隐藏文件夹

1.1 快速安装

您可以选择默认安装的方式在您的服务器上快速安装Logtash。

背景信息

日志服务提供了一个基于Logstash-2.2.2版本且集成 JRE1.8、日志服务写出插件、NSSM 2.24的安装包，部署步骤相较于 安装更加简洁，如有复杂需求可以选择自定义安装。

操作步骤

1. 下载安装包后解压缩到 C 盘。
2. 确认Logstash的启动程序路径为 `C:\logstash-2.2.2-win\bin\logstash.bat`。

1.2 Syslog



说明：

采集syslog文件，建议您使用Logtail插件，详细说明请参考*Syslog*输入源。

Logtail支持在本地配置TCP端口，接收syslog Agent通过TCP协议转发过来的syslog数据，Logtail解析接收到的数据并转发至LogHub中。

前提条件

设置使用Logtail收集日志前，您需要安装Logtail。Logtail支持Windows和Linux两大操作系统，安装方法参见 *Linux* 和*Windows*。

步骤 1 在日志服务管理控制台创建Logtail syslog配置

1. 在日志服务云控制台单击目标项目，进入**Logstore**列表。
2. 选择目标Logstore，并单击**数据接入向导**图标，进入数据接入流程。
3. 选择数据源类型。

   单击**自定义数据**中的**Syslog**，并单击下一步。
4. 指定 Logtail配置的名称。

   配置名称只能包含小写字母、数字、连字符（-）和下划线（_），且必须以小写字母和数字开头和结尾，长度为 3~63 字节。

> **说明：**
> 配置名称设置后不可修改。

5. 填写**Tag**设置。

   如何设置Tag，请参考*Syslog*-采集参考。

   图 **1-1:** 设置**Tag**



6. 酌情配置高级选项。

   请选择是否打开**本地缓存**。当日志服务不可用时，日志可以缓存在机器本地目录，服务恢复后进行续传。默认开启缓存，最大缓存值1GB。

7. 根据页面提示，应用Logtail配置到机器组。

   确认勾选所需的机器组并单击 **应用到机器组** 将配置应用到机器组。

   如果您还未创建机器组，需要先创建一个机器组。有关如何创建机器组，参见 创建*IP*地址机器组。

图 1-2: 应用到机器组



步骤 2 配置Logtail使协议生效

从机器Logtail安装目录下找到 `ilogtail_config.json`，一般在 `/usr/local/ilogtail/`目录下面。根据需求修改和syslog相关的配置。

1. 确认syslog功能已开启。

true表示syslog功能处于打开状态，false表示关闭状态。

```
"streamlog_open" : true
```

2. 配置syslog用于接收日志的内存池大小。程序启动时会一次性申请指定大小的内存，请根据机器内存大小以及实际需求填写，单位是MB。

```
"streamlog_pool_size_in_mb" : 50
```

3. 配置缓冲区大小。需要配置Logtail每次调用socket io rcv 接口使用的缓冲区大小，单位是byte。

```
"streamlog_rcv_size_each_call" : 1024
```

4. 配置日志syslog格式。

```
"streamlog_formats":[]
```

5. 配置TCP绑定地址和端口。需要配置Logtail用于接收syslog日志的TCP绑定地址和端口，默认是绑定0.0.0.0下的11111端口。

```
"streamlog_tcp_addr" : "0.0.0.0",
"streamlog_tcp_port" : 11111
```

6. 配置完成后重启Logtail。重启Logtail要执行以下命令关闭Logtail客户端，并再次打开。

```
sudo /etc/init.d/ilogtaild stop
sudo /etc/init.d/ilogtaild start
```

步骤 3 安装rsyslog并修改配置

如果机器已经安装rsyslog，忽略这一步。

1. 安装rsyslog。

   安装方法请参见：

   - *Ubuntu* 安装方法
   - *Debian* 安装方法
   - *RHEL/CENTOS* 安装方法

2. 修改配置。

   在 `/etc/rsyslog.conf` 中根据需要修改配置，例如：

```
$WorkDirectory /var/spool/rsyslog # where to place spool files
$ActionQueueFileName fwdRule1 # unique name prefix for spool files
$ActionQueueMaxDiskSpace 1g # 1gb space limit (use as much as
possible)
$ActionQueueSaveOnShutdown on # save messages to disk on shutdown
$ActionQueueType LinkedList # run asynchronously
```

```
$ActionResumeRetryCount -1 # infinite retries if host is down
# 定义日志数据的字段
$template ALI_LOG_FMT,"0.1 sys_tag %timegenerated:::date-unixtimest
amp% %fromhost-ip% %hostname% %pri-text% %protocol-version% %app-
name% %procid% %msgid% %msg:::drop-last-lf%\n"
*.* @@10.101.166.173:11111;ALI_LOG_FMT
```



说明：

模板 `ALI_LOG_FMT` 中第二个域的值是 `sys_tag`，这个取值必须和步骤 1 中创建的一致，这个配置的含义是将本机接收到的所有（`\*.\*`）syslog 日志按照 `ALI_LOG_FMT`格式化，使用 TCP 协议转发到 10.101.166.173:11111。机器 10.101.166.173 必须在步骤 1 中的机器组中并且按照步骤 2 配置。

3. 启动rsyslog。

```
sudo /etc/init.d/rsyslog restart
```

启动之前请先检查机器上是否安装了其他syslog的Agent，比如 syslogd、sysklogd、syslog-ng 等，如果有的话请关闭。

上面三步完成之后就可以将机器上的syslog收集到日志服务了。

更多信息

有关syslog日志采集的更多信息以及如何格式化syslog数据，请参见*Syslog-采集参考*。

1.3 Syslog-采集参考

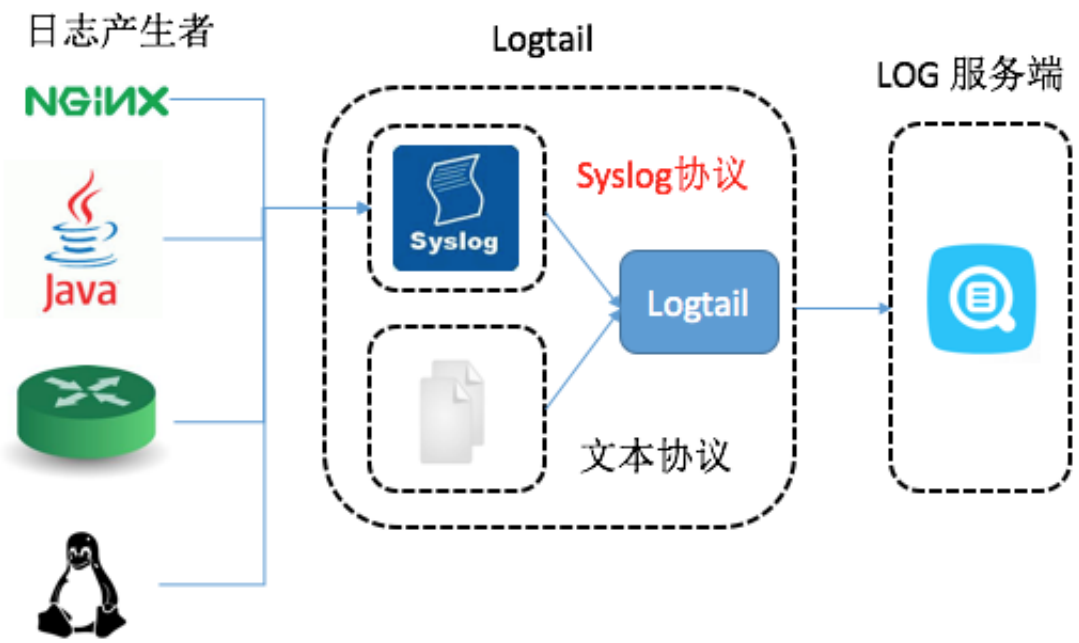


说明：

采集syslog文件，建议您使用Logtail插件，详细说明请参考*Syslog输入源*。

Logtail目前支持的接入端为syslog和文本文件，如下图所示：

图 1-3: Logtail支持的接入端



Logtail通过TCP协议支持syslog。配置Logtail采集syslog日志详细步骤请参见*Syslog*通过Logtail采集syslog日志。

syslog优势

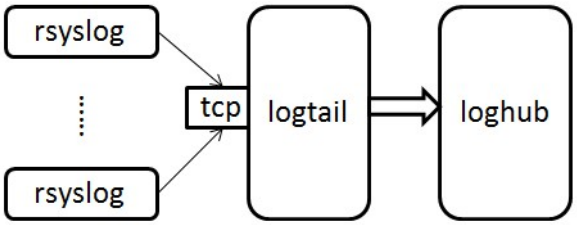
syslog概念请参考 *syslog*。

和利用文本文件相比，使用syslog时日志数据直接收集到LogHub，syslog不落盘、保密性好。免去了文件落盘和解析的代价，单机可达 80MB/S 吞吐率。

基本原理

Logtail 支持在本地配置TCP端口，接收syslog Agent转发的日志。Logtail开启TCP端口，接收rsyslog或者其他syslog Agent通过TCP协议转发过来的syslog数据，Logtail解析接收到的数据并转发到LogHub中。配置Logtail采集syslog日志过程请参见*Syslog*。Logtail、syslog、LogHub三者之间的关系如下图所示。

图 **1-4:** 基本原理



syslog日志格式

Logtail 通过 TCP 端口接收到的数据是流式的，如果要从流式的数据中解析出一条条的日志，日志格式必须满足以下条件：

- 每条日志之间使用换行符（`\n`）分隔，一条日志内部不可以出现换行符。
- 日志内部除了消息正文可以包含空格，其他字段不可以包含空格。

syslog日志格式如下：

```
$version $tag $unixtimestamp $ip [$user-defined-field-1 $user-defined-field-2 $user-defined-field-n] $msg\n"
```

各个字段含义为：

日志字段	含义
version	该日志格式的版本号，Logtail使用该版本号解析user-defined-field 字段。
tag	数据标签，用于寻找Project或Logstore，不可以包含空格和换行符。
unixtimestamp：	该条日志的时间戳。
ip	该条日志的对应的机器IP，如果日志中的该字段是 127.0.0.1，最终发往服务端的日志数据中该字段会被替换成TCP socket的对端地址。
user-defined-field	用户自定义字段，中括号表示是可选字段，可以有 0 个或多个，不可以包含空格和换行符。
msg	日志消息正文，不可以包含换行符，末尾的 `\n` 表示换行符。

以下示例日志即为满足格式要求的日志：

```
2.1 streamlog_tag 1455776661 10.101.166.127 ERROR com.alibaba.
streamlog.App.main(App.java:17) connection refused, retry
```

另外，不仅 syslog 日志可以接入Logtail，任何日志工具只要能满足以下条件，都可以接入：

- 可以将日志格式化，格式化之后的日志格式满足格式要求。
- 可以通过 TCP 协议将日志 append 到远端。

Logtail解析syslog日志规则

Logtail 需要增加配置以解析syslog日志。例如：

```
"streamlog_formats":
[
    {"version": "2.1", "fields": ["level", "method"]},
    {"version": "2.2", "fields": []},
    {"version": "2.3", "fields": ["pri-text", "app-name", "syslogtag
"]}
]
```

Logtail通过读取到的`version`字段到`streamlog_formats`中找到对应的`user-defined`字段的格式，应用该配置，上面的日志样例`version`字段为 2.1，包含两个自定义字段`level`和`method`，因此日志样例将被解析为如下格式：

```
{
    "source": "10.101.166.127",
    "time": 1455776661,
    "level": "ERROR",
    "method": "com.alibaba.streamlog.App.main(App.java:17)",
    "msg": "connection refused, retry"
}
```

`version`用于解析`user-defined`字段，tag用于寻找数据将要被发送到的Project或Logstore，这两个字段不会作为日志内容发送到阿里云日志服务。另外，Logtail预定义了一些日志格式，这些内置的格式都使用 0.1、1.1 这样以“0.”、“1.”开头的`version`值，所以用户自定义`version`不可以以“0.”、“1.”开头。

常见日志工具接入 Logtail syslog log

- **log4j**

  — 引入 log4j 库。

  ```
  <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-api</artifactId>
        <version>2.5</version>
    </dependency>
  ```

```
  <dependency>
      <groupId>org.apache.logging.log4j</groupId>
      <artifactId>log4j-core</artifactId>
      <version>2.5</version>
  </dependency>
```

- 程序中引入 log4j 配置文件 `log4j_aliyun.xml`。

```
 <?xml version="1.0" encoding="UTF-8"?>
  <configuration status="OFF">
      <appenders>
          <Socket name="StreamLog" protocol="TCP" host="10.101.166
.173" port="11111">
              <PatternLayout pattern="%X{version} %X{tag} %d{UNIX}
 %X{ip} %-5p %l %enc{%m}%n" />
          </Socket>
      </appenders>
      <loggers>
          <root level="trace">
              <appender-ref ref="StreamLog" />
          </root>
      </loggers>
  </configuration>
```

其中 10.101.166.173:11111 是 Logtail 所在机器的地址。

- 程序中设置 ThreadContext。

```
package com.alibaba.streamlog;
  import org.apache.logging.log4j.LogManager;
  import org.apache.logging.log4j.Logger;
  import org.apache.logging.log4j.ThreadContext;
  public class App
  {
      private static Logger logger = LogManager.getLogger(App.
class);
      public static void main( String[] args ) throws Interrupte
dException
      {
          ThreadContext.put("version", "2.1");
          ThreadContext.put("tag", "streamlog_tag");
          ThreadContext.put("ip", "127.0.0.1");
          while(true)
          {
              logger.error("hello world");
              Thread.sleep(1000);
          }
          //ThreadContext.clearAll();
      }
  }
```

- **tengine**

  tengine 可以通过 syslog 接入 ilogtail。

  tengine 使用 `ngx_http_log_module`模块将日志打入本地 syslog agent，在本地 syslog agent 中 forward 到 rsyslog。

tengine 配置 syslog 请参考：*tengine配置syslog*

示例：

以 user 类型和 info 级别将 access log 发送给本机 Unix dgram(/dev/log)，并设置应用标记为 nginx。

```
access_log syslog:user:info:/var/log/nginx.sock:nginx
```

rsyslog 配置：

```
module(load="imuxsock") # needs to be done just once
input(type="imuxsock" Socket="/var/log/nginx.sock" CreatePath="on")
$template ALI_LOG_FMT,"2.3 streamlog_tag %timegenerated:::date-
unixtimestamp% %fromhost-ip% %pri-text% %app-name% %syslogtag% %msg
:::drop-last-lf%\n"
if $syslogtag == 'nginx' then @@10.101.166.173:11111;ALI_LOG_FMT
```

- **nginx**

以收集 nginx accesslog 为例。

access log 配置：

```
access_log syslog:server=unix:/var/log/nginx.sock,nohostname,tag=
nginx;
```

rsyslog 配置：

```
module(load="imuxsock") # needs to be done just once
input(type="imuxsock" Socket="/var/log/nginx.sock" CreatePath="on")
$template ALI_LOG_FMT,"2.3 streamlog_tag %timegenerated:::date-
unixtimestamp% %fromhost-ip% %pri-text% %app-name% %syslogtag% %msg
:::drop-last-lf%\n"
if $syslogtag == 'nginx' then @@10.101.166.173:11111;ALI_LOG_FMT
```

参考：*http://nginx.org/en/docs/syslog.html*

- **Python Syslog**

示例：

```
import logging
import logging.handlers
logger = logging.getLogger('myLogger')
logger.setLevel(logging.INFO)
#add handler to the logger using unix domain socket '/dev/log'
handler = logging.handlers.SysLogHandler('/dev/log')
#add formatter to the handler
formatter = logging.Formatter('Python: { "loggerName":"%(name)s",
 "asciTime":"%(asctime)s", "pathName":"%(pathname)s", "logRecordC
reationTime":"%(created)f", "functionName":"%(funcName)s", "levelNo
":"%(levelno)s", "lineNo":"%(lineno)d", "time":"%(msecs)d", "
levelName":"%(levelname)s", "message":"%(message)s"}')
```

```
handler.formatter = formatter
logger.addHandler(handler)
logger.info("Test Message")
```

1.4 实时分析

2 准备工作

2.1 准备流程

日志服务为您提供多种方式的日志收集服务，您可以通过日志服务采集ECS日志、本地服务器日志、IoT设备日志和其他云产品日志等。

在您开始使用日志服务时，首先需要进行以下准备工作。

操作步骤

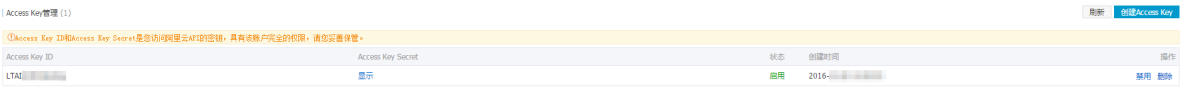
1. 开通日志服务。

   使用注册成功的阿里云账号登录日志服务产品页，单击立即购买。系统自动跳转到购买页面。单击立即开通之后，购买成功。您将自动跳转到日志服务控制台。

2. 开启Access Key（适用于API/SDK操作）。

   Access Key是您通过API/SDK操作日志服务的必要条件。开始使用日志服务之前，您可以选择创建密钥对。

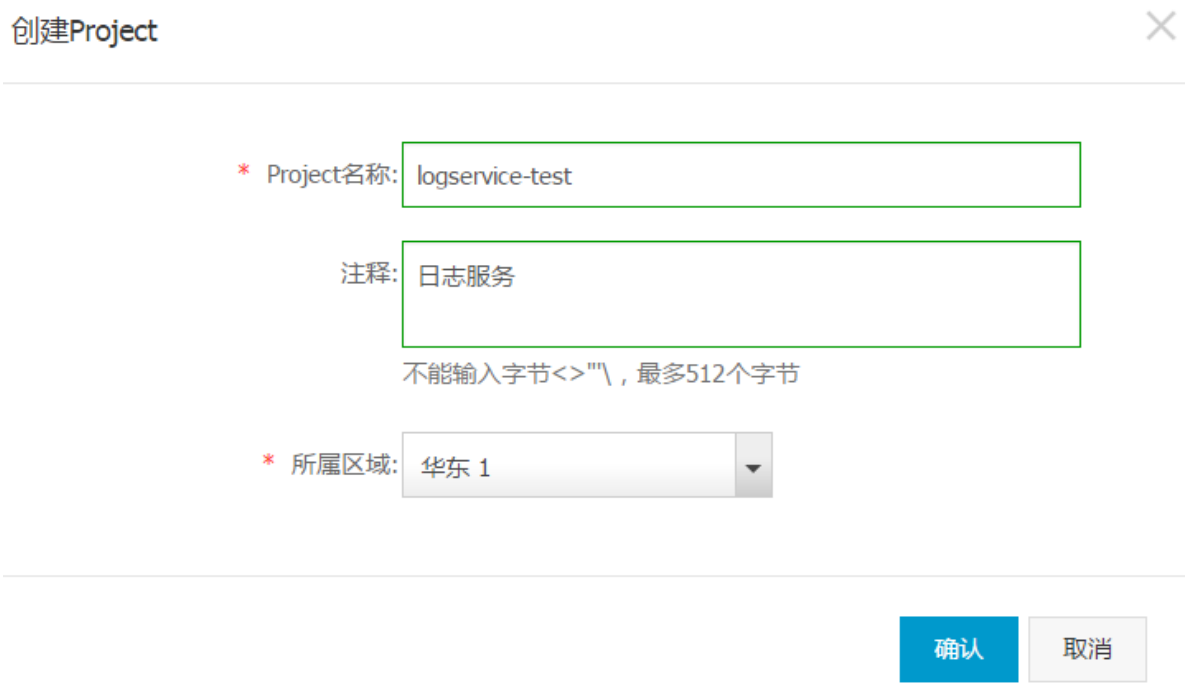
   在 日志服务管理控制台，将鼠标移至页面右上角您的用户名上方，在显示的菜单中单击**accesskeys** 。在弹出的确认对话框中单击继续使用**AccessKey** 以进入 **Access Key**管理页面。创建密钥对（Access Key），确认状态已设置为启用。

3. 创建Project。

   当您第一次进入日志服务管理控制台，系统会提示您创建一个项目（Project）。您也可以通过单击右上角的 创建**Project** 进行操作。

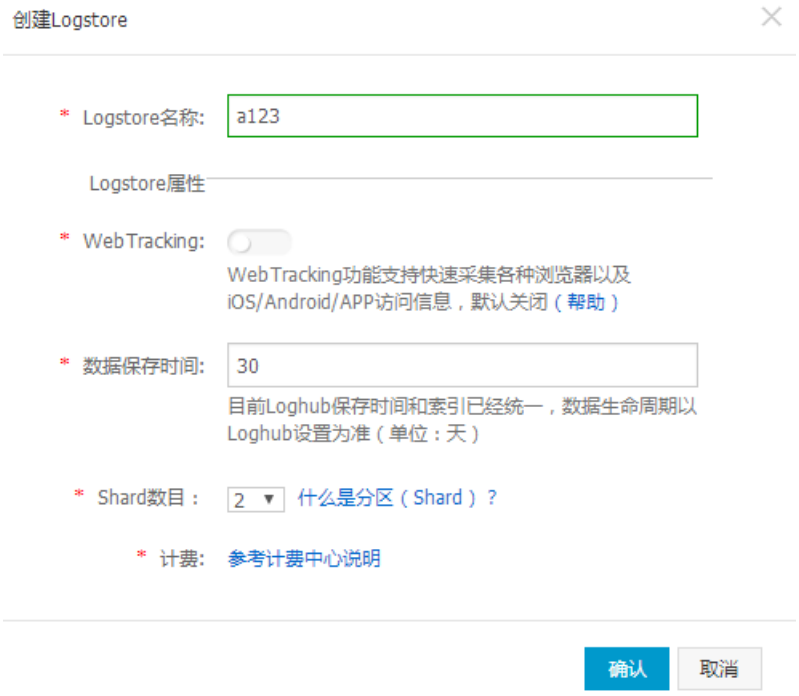
   您还可以修改Project的注释、删除Project，详情请参考操作*Project*。



4. 创建Logstore。

   在Project创建完成的同时，系统会提示您创建一个日志库。您也可以进入该Project，通过单击右上角的 创建 进行操作。创建Logstore需要指定如何使用这些日志。

   您还可以对Logstore执行修改、删除等操作，详情请参考操作*Logstore*。



5. 操作Shard（可选）。

当您新建Logstore时，可以根据您的日志量和产生速度选择Shard个数，您也可以在修改Logstore时，通过合并拆分等操作，修改Shard的个数。

Shard的合并、拆分等操作，请参考操作*Shard*。

6. 访问控制授权（可选）。

   如您需要收集云产品的日志，或将日志服务的数据投递至OSS等产品进行存储分析，您需要赋予日志服务或其他云产品相应的权限。

   或者您需要使用子账号操作日志服务，您也需要通过RAM对子账号授权。

   关于授权策略及授权步骤，请参考简介。

2.2 操作Project

您可以通过日志服务管理控制台对项目（Project）进行创建和删除操作。

创建Project



说明：

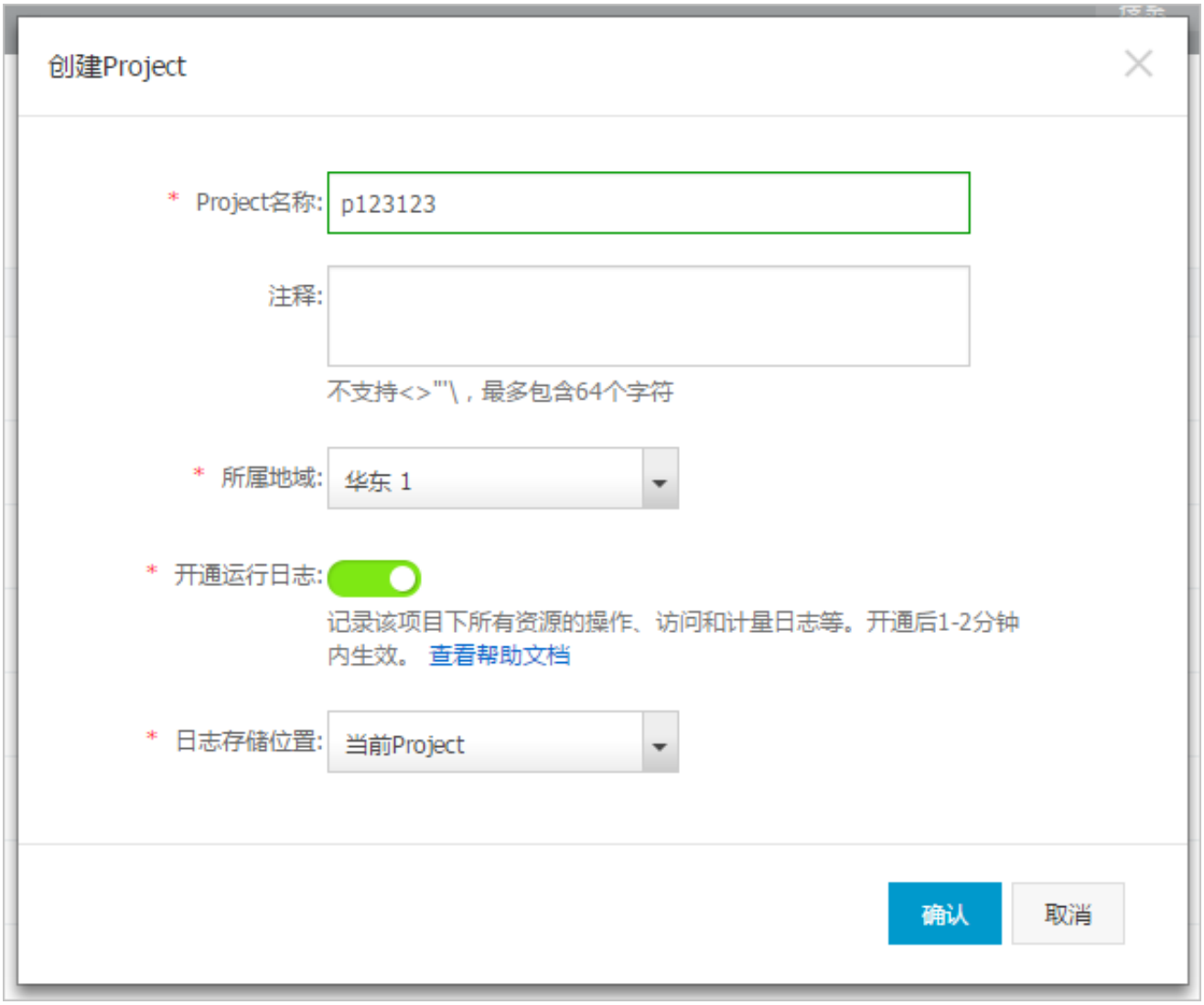
- 目前，日志服务仅提供控制台方式创建Project。
- Project的名字需要全局唯一（在所有阿里云Region内）。如果您选择的Project名称已经被别人使用，您会收到页面提醒信息“**Project XXX already exists**”，请您更换一个Project名称重试。
- Project创建时需要指定所在的阿里云Region。请根据需要收集的日志来源和其他实际情况选择合适的阿里云Region。如果您需要收集来自阿里云ECS的日志，建议在ECS相同的Region 创建Project。这样可以加快日志收集速度，还可以使用阿里云内网收集日志，不占用 ECS 虚拟机公网带宽。
- Project一旦创建完成则无法改变其所属地域，且日志服务不支持Project的迁移，所以请谨慎选择Project的所属Region。
- 一个阿里云账户在所有阿里云Region最多可创建50个Project。

操作步骤

1. 登录日志服务管理控制台。
2. 单击右上角的 创建**Project**。
3. 填写 **Project**名称 和 所属地域，确认是否开通运行日志功能，并单击 确认。

配置项	说明
Project名称	项目名称只能包含小写字母、数字和连字符（-），且必须以小写字母和数字开头和结尾，长度为 3~63 个字节。 **说明：** 项目名称创建后不能修改。
注释	Project的简单注释，配置后将显示在**Project**列表页面。创建Project完成后可以在**Project**列表页面单击修改来修改注释。 **说明：** 注释内容长度为0~64个字节，不支持字符<>"'\。
所属地域	您需要为每个项目指定阿里云的一个地域（Region），且创建后就不能修改地域，也不能在多个地域间迁移项目。
开通运行日志	运行日志功能的开关，开启该功能后日志服务将记录该项目下所有资源的操作、访问和计量日志等。开通后1-2分钟内生效。
日志存储位置	如果选择开启运行日志功能，您需要选择日志的存储位置。可以设置为： • 当前Project。 • 同一地域下的其他Project。

图 2-1: 创建Project



修改Project

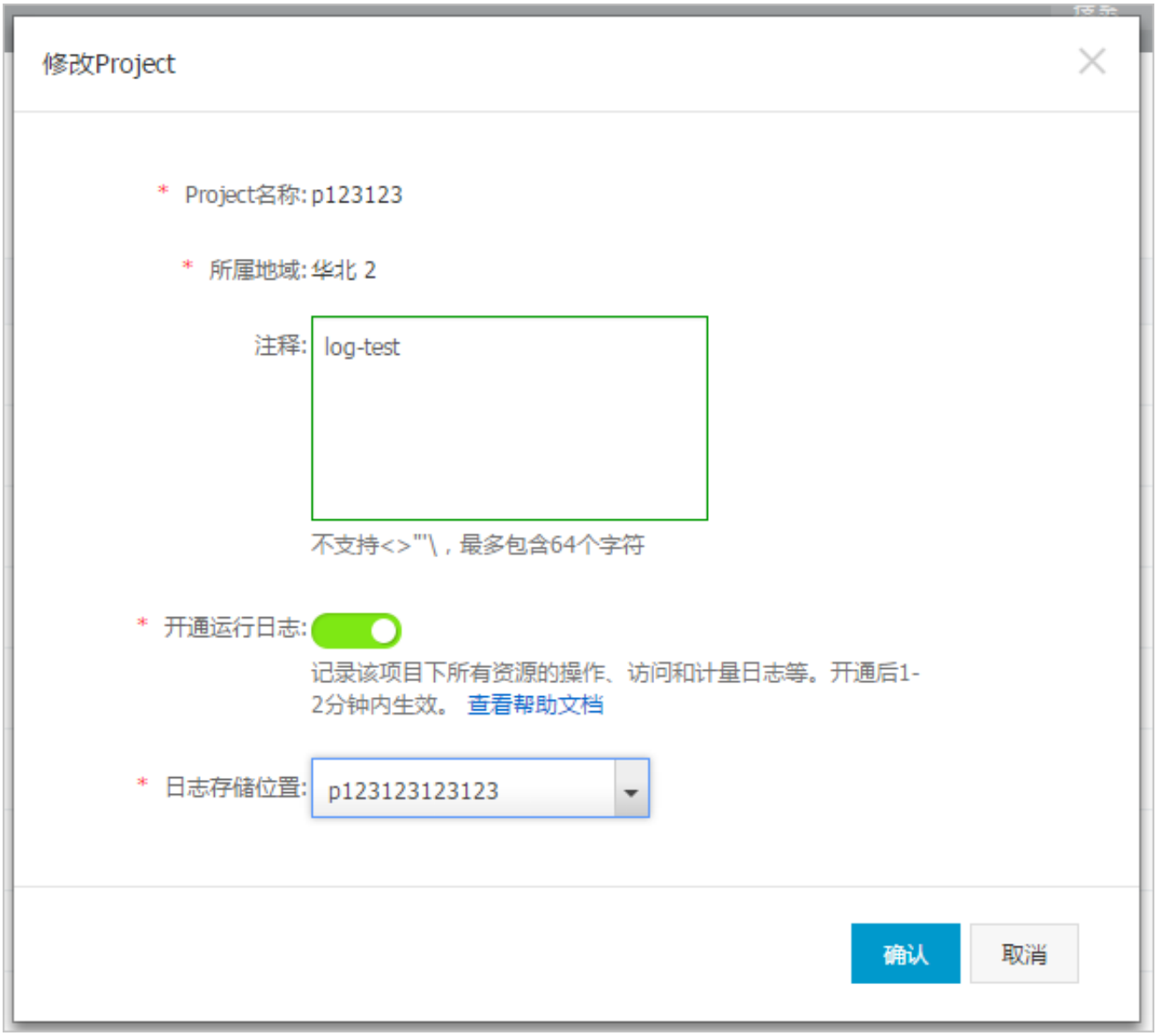
如果您需要修改Project的注释信息、开通或关闭记录日志功能或修改日志服务日志的存储位置，可以通过修改Project来操作。

1. 登录日志服务控制台。
2. 找到需要修改配置的Project。您可以在**Project**列表左上角搜索框内输入Project名称并单击搜索进行查询。
3. 单击操作列的修改。
4. 在弹出对话框中修改Project配置。

> 说明：

不支持修改Project名称和所属地域。

图 **2-2:** 修改**Project**



删除**Project**

在某些情况下（例如关闭日志服务，销毁Project 的所有日志等），您可能需要删除整个Project。

说明：

当您的Project被删除后，其管理的所有日志数据及配置信息都会永久释放，不可恢复。所以，在删除Project 前请慎重确认，避免数据丢失。

1. 在Project列表中，选择需要删除的项目。

2. 单击右侧的 删除。
3. 在弹出的对话框中，单击 获取验证码，输入您收到的校验码并单击确认 。



2.3 操作Logstore

日志库（Logstore）是创建在项目（Project）下的资源集合，Logstore中的所有数据都来自于同一个数据源。

收集到的日志数据的查询、分析、投递均以Logstore为单位。您可以对Logstore进行以下操作：

- 创建*Logstore*
- 修改*Logstore*配置
- 删除*Logstore*

创建Logstore

> 说明：
>
> - 任何一个 Logstore 必须在某一个 Project 下创建。
> - 每个日志服务项目可创建最多200个日志库。
> - Logstore 名称在其所属Project内必须唯一。
> - 数据保存时间创建后还可以进行修改。您可以在 Logstore 页面，在操作列下单击操作 > 修改，修改 数据保存时间 并单击修改 ，然后关闭对话框即可。

1. 在 **Project**列表 页面，单击项目的名称，然后单击创建创建日志库。

   或者在创建完项目后，根据系统提示创建日志库，单击 创建。
2. 填写日志库的配置信息并单击 确定。

配置项	说明
Logstore名称	日志库名称须由小写字母、数字、连字符（-）和下划线（_）组成，且以小写字母或者数字开头和结尾，长度为3-63字节。Logstore 名称在其所属项目内必须唯一。  说明： 日志库名称创建后不能修改。
WebTracking	确认是否开启WebTracking功能。WebTracking功能支持从HTML、H5、iOS、或Android平台收集日志数据到日志服务。默认关闭。
永久保存	确认是否开启**永久保存**功能，默认为开启状态。 日志服务支持永久保存采集到的日志数据，功能。您也可以关闭该功能，并自定义设置**数据保存时间**。
数据保存时间	采集到日志服务中的日志在日志库中的保存时间，单位为天。可以设置为1~3000天。超过该时间后，日志会被删除。 如果您关闭了**永久保存**功能，则需要自定义设置**数据保存时间**。
Shard数目	日志库的分区数量，每个Logstore可以创建1~10个分区。每个Project中可以创建最多200个分区。
自动分裂Shard	确认是否开启**自动分裂Shard**功能，默认为开启状态。 当数据量超过已有分区（Shard）服务能力后，开启自动分裂功能可自动根据数据量增加分区数量。关于自动分裂Shard的详细说明，请参考操作*Shard*。
最大分裂数	最大Shard自动分裂后的最大数目，最大可支持自动分裂至64个分区。 如果您开启了**自动分裂Shard**功能，则需要设置**最大分裂数**。
记录外网IP	开启后，服务端接收到日志数据之后，自动把以下信息添加到日志的Tag字段中。 • `__client_ip__`：日志来源设备的公网IP地址。 • `__receive_time__`：日志到达服务端的时间，格式为Unix时间戳。



修改Logstore配置

创建日志库以后，您还可以在需要的时候修改日志库的配置。

1. 登录日志服务控制台。
2. 选择所需的项目，单击项目名称。
3. 在 **Logstore列表** 页面，选择所需的日志库并单击操作列下的 修改。
4. 在弹出的对话框中修改日志库的配置并关闭对话框。



删除Logstore

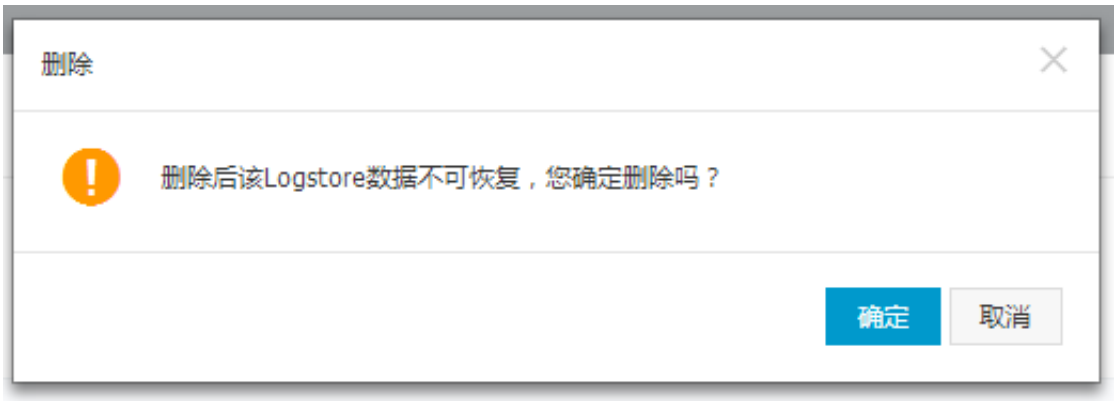
在某些情况下（如希望废弃某个 Logstore），您可能需要删除指定的 Logstore。日志服务允许您在控制台上删除 Logstore。

说明：

- 一旦 Logstore 删除，其存储的日志数据将会被永久丢失，不可恢复，请谨慎操作。

- 删除指定 Logstore 前必须删除其对应的所有 Logtail 配置。
- 如果该 Logstore 上还启用了日志投递的消费模式，则不保证删除前 Logstore 里的所有数据都会成功投递到 MaxCompute 中。如果您需要保证被删除的 Logstore 内所有数据都能投递到 MaxCompute，请按照如下流程操作：
  1. 删除前先停止向该 Logstore 写入新日志。
  2. 确认 Logstore 里的所有日志数据都成功导入到 MaxCompute。
  3. 删除 Logstore。

1. 登录日志服务控制台。
2. 选择所需的项目，单击项目名称。
3. 在 **Logstore列表** 页面，选择要删除的日志库并单击右侧的 **删除**。
4. 在弹出的确认对话框中，单击 **确定**。



2.4 操作Shard

Logstore读写日志必定保存在某一个分区（Shard）上。每个日志库（Logstore）分若干个分区，您在创建Logstore时需要为该Logstore设置Shard的数量，创建完成后还可以分裂或合并Shard，以达到增加或减少Shard的目的。

对于已存在的Shard，您可以进行以下操作：

- 分裂*Shard*
- 自动分裂*Shard*
- 合并*Shard*
- 删除*Shard*

分裂Shard

每个分区（Shard）能够处理5M/s的数据写入和10M/s的数据读取，当数据流量超过分区服务能力时，建议您及时增加分区。扩容分区通过分裂（split）操作完成。

> **说明：**
> 您也可以通过日志服务命令行工具CLI一次性分裂Shard到指定数量，详细说明请参考使用*CLI*配置*Shard*。

使用指南

在分裂分区时，需要指定一个处于readwrite状态的ShardId和一个MD5。MD5要求必须大于分区的BeginKey并且小于EndKey。

分裂操作可以从一个分区中分裂出另外两个分区，即分裂后分区数量增加2。在分裂完成后，被指定分裂的原分区状态由readwrite变为readonly，数据仍然可以被消费，但不可写入新数据。两个新生成的分区状态为readwrite，排列在原有分区之后，且两个分区的MD5范围覆盖了原来分区的范围。

1. 登录日志服务管理控制台。
2. 选择所需的项目，单击项目名称。
3. 在**Logstore列表**页面，选择所需的日志库并单击操作列下的**修改**。
4. 选择要分裂的分区，单击右侧的**分裂**。



5. 单击**确认** 并关闭对话框。

   分裂操作完成后，原分区变为readonly状态，两个新生成的分区的MD5范围覆盖了原来分区的范围。



自动分裂Shard

使用指南

除手动分裂Shard之外，日志服务还支持自动分裂Shard功能。您可以在创建或修改Logstore时开启该功能，并设定Shard自动分裂后的最大数目。

开启自动分裂Shard功能后，满足以下条件时，Shard会自动分裂：

1. 数据量超出当前已有Shard的服务能力，且持续5分钟以上。
2. Logstore中readwrite状态的Shard数目未超过设定的最大shard总数。

> 说明：
> 最近15分钟内分裂出来的新Shard不会自动分裂。

图 2-3: 自动分裂Shard



配置项	说明
自动分裂shard	Shard自动分裂功能开关。开启该功能后，满足条件的Shard会在数据量超出Shard服务能力时自动分裂。
最大分裂数	Shard自动分裂后的最大数目。开启自动分裂Shard功能后，最大可支持自动分裂至64个分区。

合并Shard

您可以通过合并（merge）操作缩容分区。合并操作可以将指定分区与其右侧相邻分区的范围合并，并将此范围赋予新生成的readwrite分区，两个被合并的原分区变为readonly状态。

使用指南

在合并操作时，必须指定一个处于readwrite状态的分区，指定的分区不能是最后一个readwrite分区。服务端会自动找到所指定分区的右侧相邻分区，并将两个分区范围合并。在合并完成后，所指定的分区和其右侧相邻分区变成只读（readonly）状态，数据仍然可以被消费，但不能写入新数据。同时新生成一个 readwrite 状态的分区，新分区的MD5范围覆盖了原来两个分区的范围。

操作步骤

1. 登录日志服务管理控制台。
2. 选择所需的项目，单击项目名称。
3. 在**Logstore**列表页面，选择所需的日志库并单击操作列下的修改。
4. 选择要合并的分区，单击右侧的合并 并关闭对话框即可。



在合并完成后，所指定的分区和其右侧相邻分区变成只读（readonly）状态，新生成的readwrite分区的MD5范围覆盖了原来两个分区的范围。



删除Shard

Logstore的生命周期即数据保存时间支持设置为1~3000天，分区及分区中的日志数据在超出该时间后会自动删除。readonly 分区不参与计费。您也可以将某个Logstore中的日志数据设置为永久保存，这部分Shard和日志数据将永远不会自动删除。修改日志数据自动保存时间请参考操作*Logstore*。

您也可以通过删除Logstore的方式删除整个日志库中的所有分区。

3 数据采集

3.1 采集方式

LogHub 支持客户端、网页、协议、SDK/API等多种日志无损采集方式，所有采集方式均基于Restful API实现，除此之外您也可以通过API/SDK实现新的采集方式。

采集来源

类别	来源	接入方式	更多
应用	程序输出	*Logtail*	案例
	访问日志	*Logtail*	分析*Nginx*日志
	链路跟踪	Jaeger Collector，*Logtail*	-
语言	Java	*SDK*，*Java Producer Library*	-
	Log4J Appender	*1.x*，*2.x*	-
	LogBack Appender	*LogBack*	-
	C	*Native*	-
	Python	*Python*	-
	Python Logging	*Python Logging Handler*	-
	PHP	*PHP*	-
	C#	*Go*	-
	Go	*Go*	-
	NodeJS	*NodeJs*	-
	JS	*JS/Web Tracking*	-
OS	Linux	*Logtail*	-
	Windows	*Logtail*	-
	Mac/Unix	*Native C*	-
	Docker文件	*Logtail* 文件采集	-
	Docker输出	*Logtail* 容器输出	-
数据库	MySQL Binlog	*MySQL Binlog*方式	-

类别	来源	接入方式	更多
	JDBC Select	*JDBC*查询结果	-
移动端	iOS/Android	*Android SDK*，*Android SDK*	-
	网页	*JS/Web Tracking*	-
	智能IoT	*C Producer Library*	案例
其他	HTTP 轮询	*Logtail HTTP*	*Nginx*监控
	Syslog	*Logtail*插件-*syslog*输入源	-
云产品	云服务器（ECS）	*Logtail*采集简介	-
	容器服务（CS）	文本，*FluentBit*（客户提供）	文本，输出
	对象存储（OSS）	产品页开通	说明
	负载均衡（SLB）	产品页开通	说明
	安骑士	产品页开通	说明
	函数计算（FC）	产品页开通	说明
	API网关（API）	产品页开通	说明
	消息服务（MNS）	产品页开通	说明
	MaxCompute	*DataWorks*导入数据	-
	态势感知	请提工单，申请白名单。	**说明：** 仅态势感知（企业版）支持日志功能，同时需要开通日志检索。
	内容分发（CDN）	请提工单，申请白名单。	-
	DDoS	请提工单，申请白名单。	简介
	ActionTrail	产品页开通	简介
第三方	Logstash	*Logstash*	-

网络与接入点选择

日志服务在各 Region 提供 服务入口，每个 Region 提供两种网络接入方式：

- 内网（经典网）/私网（专有网络 VPC）：本 Region 内服务访问，带宽链路质量最好（推荐）。
- 公网（经典网）：可以被任意访问，访问速度取决于链路质量、传输安全保障建议使用 HTTPS。

常见问题

- Q：专线方式接入应如何选择网络？

  A：请选择内网/私网接入点。
- Q：采集公网数据时能否采集公网IP？

  A：参考操作*Logstore*，开通记录外网IP功能。
- Q：把另一个**Region A**上的**ECS**日志采集到本**Region B**下日志服务**Project**中，应如何选择网络？

  A：在Region A ECS上安装 Region B公网版本Logtail，进行公网传输。其他情况下如何选择网络，请参考选择网络。
- Q：如何快速判断能否连接？

  A：运行以下命令，如果有返回信息则表示可以联通。

  ```
  curl $myproject.cn-hangzhou.log.aliyuncs.com
  ```

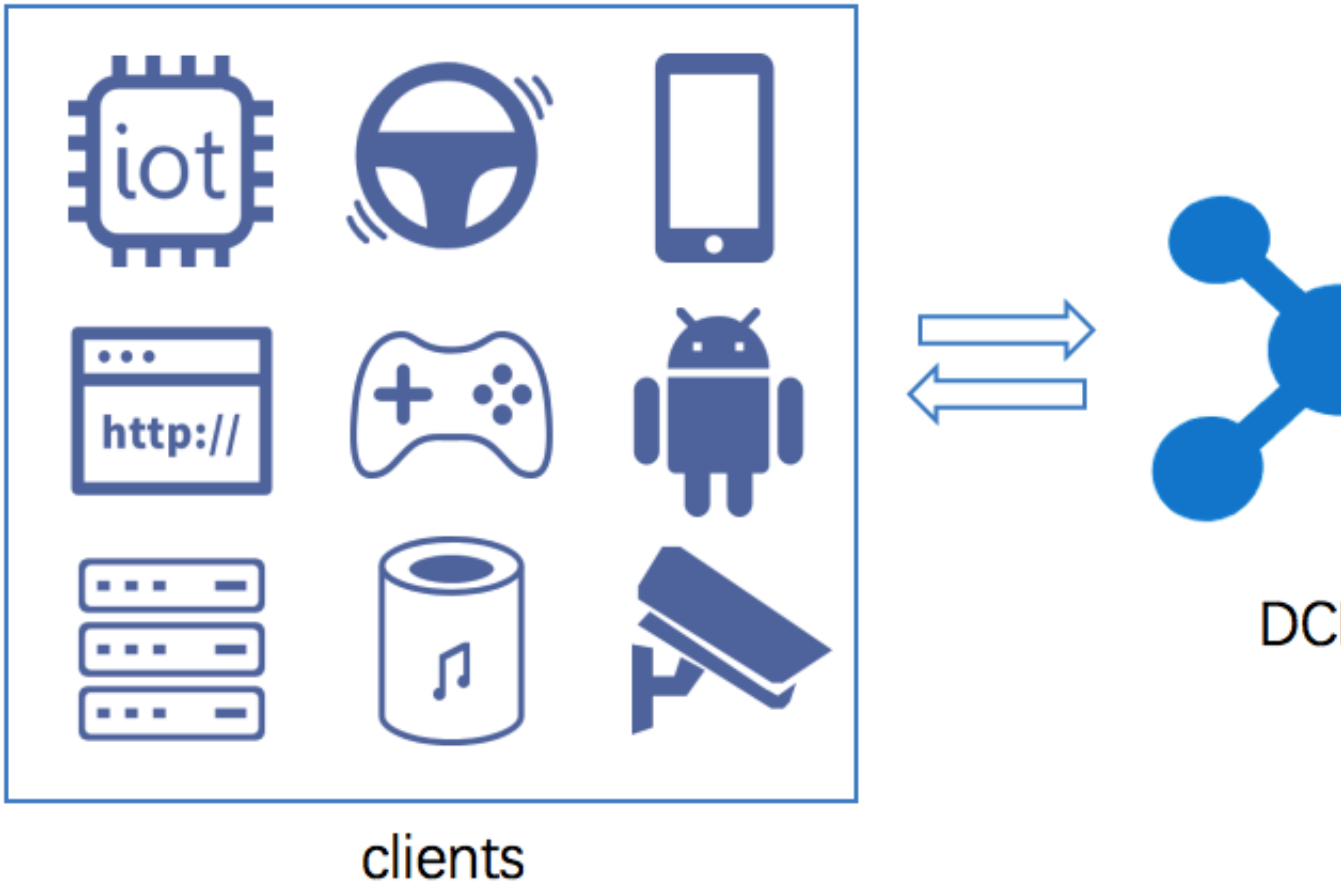
  其中，`$myproject`表示Project名，`cn-hangzhou.log.aliuncs.com`表示访问接入点。

3.2 采集加速

3.2.1 简介

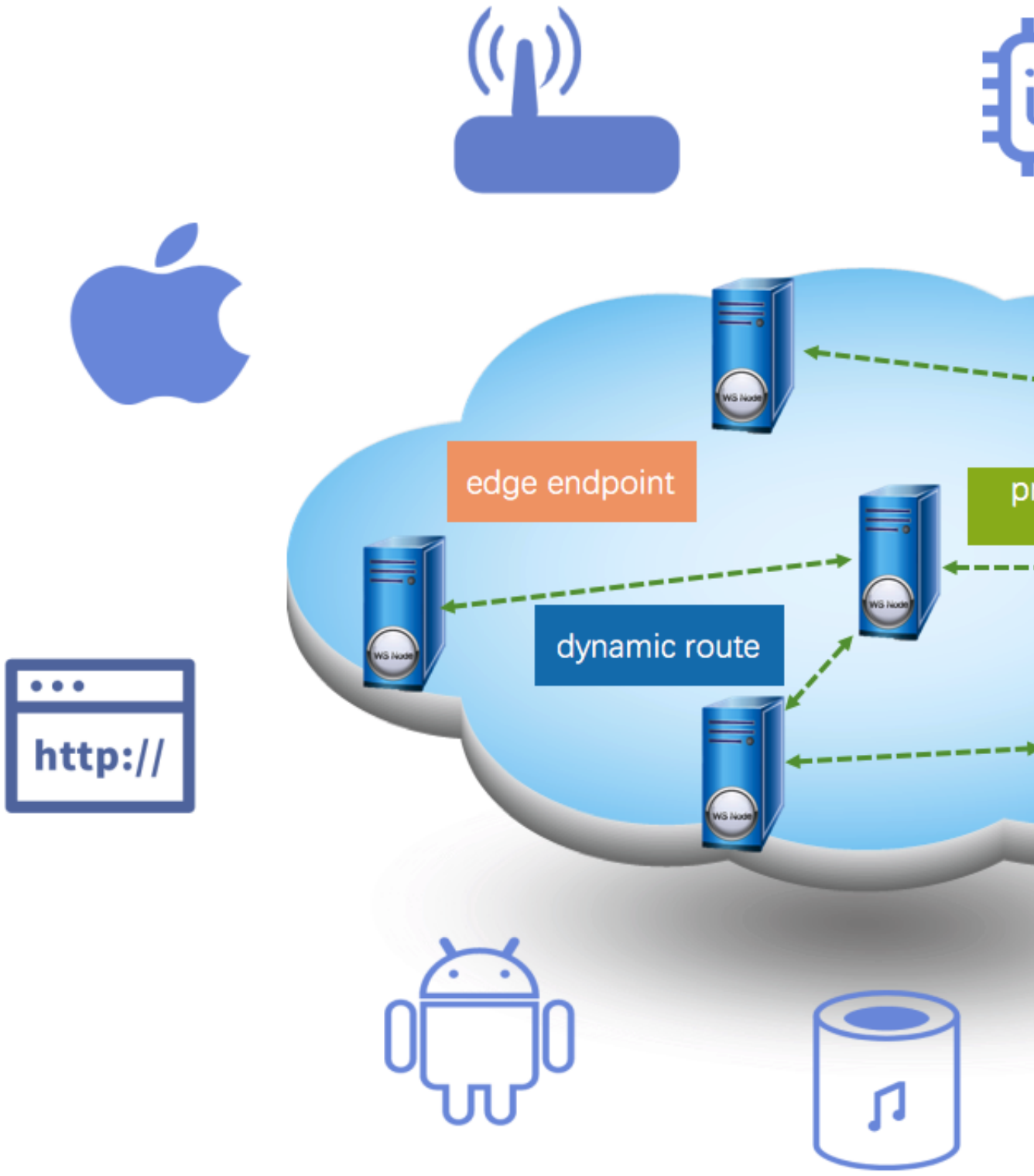
日志服务在VPC和公网基础上，新增 全球加速公网 的网络类型。相较于普通的公网访问，全球加速公网在延迟和稳定性两点上具备显著优势，适用于对数据采集、消费延时、可靠性要求较高的场景。日志服务全球加速功能依赖于阿里云全站加速产品提供的加速环境，解决了跨运营商、网络不稳定、突发流量、网络拥塞等诸多因素导致的响应慢、丢包、服务不稳定的问题，提升全站性能和用户体验。

日志服务全球加速功能基于阿里云CDN硬件资源，优化来自手机、IOT设备、智能设备、自建IDC、其他云服务器等多种形式数据源的日志采集、数据传输的稳定性。



技术原理

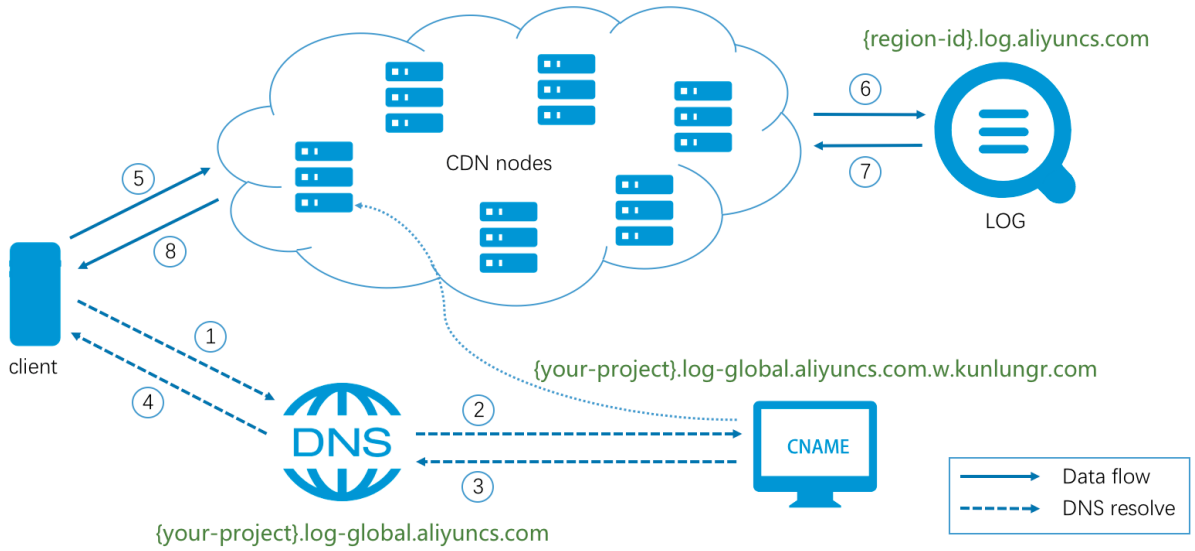
日志服务全球加速功能基于阿里云CDN硬件资源，您的全球接入端（例如手机、IOT设备、智能设备、自建IDC、其他云服务器等）会就近接入阿里云CDN遍布全球的边缘节点，通过CDN内部高速通道路由至日志服务，相比普通公网传输，可大大降低网络延迟和抖动。



日志服务全球加速请求处理流程如上图所示，整体流程详细说明如下：

1. 用户向日志服务的加速域名*your-project*.log-global.aliyuncs.com发起日志上传、下载等请求，首先需要向公共DNS发起域名解析请求。

2. 公共DNS处的*your-project*.log-global.aliyuncs.com域名指向的是CNAME地址*your-project*.log-global.aliyuncs.com.w.kunlungr.com，此时域名请求会转发至阿里云CDN的CNAME节点。
3. CNAME节点基于阿里云CDN智能调度系统，会将最优的CDN边缘节点IP返回给公共DNS。
4. 公共DNS将最终解析到的IP地址返回给客户端。
5. 客户端根据获取的IP地址向服务端发起请求。
6. CDN边缘节点接收到请求后，基于动态路由查找、私有传输协议将请求路由至距离日志服务服务端最近的节点，最后将请求转发到日志服务。
7. 日志服务的服务端接收到CDN节点请求后，将请求结果返回给CDN节点。
8. CDN将日志服务返回的请求结果/数据透传到客户端。



计费方式

日志服务全球加速费用包括：

- 访问日志服务产生的费用

  日志服务的访问费用和普通公网价格一致，为按量计费模式，并提供一定的免费额度。详细计费说明请参考计费方式。
- 全站加速的服务费用

  阿里云全站加速云产品的费用说明请参考全站加速计费方式。

应用场景

- 广告

广告浏览、点击相关的日志数据对于广告计费极其重要，而且广告载体包括遍布全球的移动端嵌入、H5页面、PC端等。在某些偏远地区，公网数据传输稳定性较差，存在丢失日志的风险，可通过全球加速服务获得更稳定可靠的日志上传通道。

- 游戏

  游戏行业对官网、登录服务、售卖服务、游戏服务等各个环节数据采集的性能和稳定性要求较高，尤其在手游数据采集、出海游戏数据回传等情况下，数据采集的时效性和稳定性难以保证。推荐您使用日志服务全球加速功能解决以上问题。

- 金融

  金融类相关应用对于网络的高可用性和高安全性要求较高， 对于每一笔交易、每一个用户操作相关的审计日志都要安全可靠地采集到服务端。目前移动端交易已成为主流，例如网上银行、信用卡商城、移动证券等，均可通过使用日志服务HTTPS全球加速功能实现安全、快速、稳定的日志采集。

- 物联网

  IOT设备、智能设备（智能音箱、智能手表等）会将相关的传感器数据、操作日志、关键性系统日志等采集到服务端进行数据分析。而这些设备通常会遍布全球，且周边的网络质量并不可控，推荐通过使用日志服务全球加速功能实现稳定可靠的日志采集。

加速效果

测试地域	延时ms（普通公网）	延时ms（加速）	超时占比%（普通公网）	超时占比%（加速）
杭州	152.881	128.501	0.0	0.0
欧洲	1750.738	614.227	0.5908	0.0
美国	736.614	458.340	0.0010	0.0
新加坡	567.287	277.897	0.0024	0.0
中东	2849.070	444.523	1.0168	0.0
澳大利亚	1491.864	538.403	0.014	0.0

其中，测试环境为：

- 日志服务所在Region：华北5(呼和浩特)
- 平均上传数据包大小：10KB

- 测试时间范围：1天（取平均值）
- 请求类型：HTTPS
- 请求服务器：阿里云ECS（规格1C1GB）



说明：

加速效果仅供参考。

3.2.2 开启全球加速

若您还未开启日志服务全球加速功能，可参考以下步骤开通。

前提条件

- 开通日志服务，并已创建Project和Logstore。
- 开通全站加速产品。
- 若您需要开启*HTTPS*加速，请先开启*HTTP*加速。

配置方式

为Project成功开启HTTP全球加速之后，您还可以根据需求配置Logtail、SDK等方式的全球加速功能。

1. 开启*HTTP*加速。
2. 开启Logtail、SDK等方式的全球加速。
   - HTTPS方式

     若您使用HTTPS访问日志服务，请确认已经开启HTTPS的加速功能，并参考开启*HTTPS*加速配置HTTPS加速。
   - Logtail日志采集

     安装Logtail时，请根据页面提示，选择网络类型为**全球加速**，之后您通过Logtail采集日志将获得全球加速效果。
   - SDK/Producer/Consumer

     其他通过SDK、Producer、Consumer等访问日志服务的方式均可通过替换endpoint获得加速效果，即将endpoint替换为`log-global.aliyuncs.com`。

开启HTTP加速

1. 登录全站加速控制台，单击左侧导航栏的**域名管理**，进入域名管理页面。

2. 单击左上角的添加域名，进入添加域名页面。
3. 填写加速域名等信息，并单击下一步。

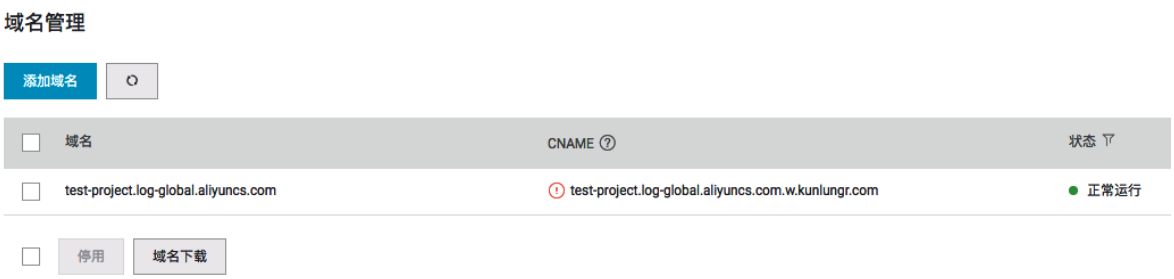
项目	内容
加速域名	*project_name*.log-global.aliyuncs.com其中*project_name*需要替换为您的Project名称。
源站类型	此处请选择源站域名。
域名	填写您Project所属Region的公网endpoint，endpoint请查看日志服务入口列表。
端口	请选择80端口，如有https加速需求可参考本文档中开通**Https**加速另行配置。
加速区域	默认不显示此配置项，且加速区域为国内加速。 如果您有全球加速的需求，请提工单到全站加速，申请白名单。 申请通过后，您可以根据需求选择加速区域。

关于添加域名的更多说明，请参考添加域名。



4. 根据页面提示，跳转到域名管理页面。

   您可以在域名管理页面中查看对应域名的**CNAME**。



5. 登录日志服务控制台，在**Project**列表中单击指定Project右侧对应的全球加速。
6. 在弹出对话框中填入加速域名对应的**CNAME**。并单击开启加速。



成功执行以上步骤即可开通日志服务全球加速功能。

开启HTTPS加速

开启HTTP加速之后，若您有HTTPS访问需求，可以通过以下步骤开通HTTPS加速功能。

1. 登录全站加速控制台，单击左侧导航栏的域名管理，进入域名管理页面。
2. 单击指定域名右侧的配置。
3. 单击左侧导航栏中的**HTTPS**配置，并在**HTTPS**证书一栏中单击修改配置，进入**HTTPS**设置页面。
4. 配置**HTTPS**安全加速和证书类型。
   - 开启**HTTPS**安全加速功能开关。
   - 证书类型请选择免费证书。



配置完毕后请勾选**同意授权阿里云申请免费证书**，并单击**确认**。

关于HTTPS设置的详细说明请参考*HTTPS*设置。

检查加速是否生效

常见问题

- **如何检查加速是否生效？**

  配置完成后，您可以通过访问您的加速域名来确认加速是否已经生效。

例如我们对*test-project* Project开启了全球加速，您可以使用curl请求加速域名，若返回以下类型输出说明加速已经生效。

```
$curl test-project.log-global.aliyuncs.com
{"Error":{"Code":"OLSInvalidMethod","Message":"The script name is
invalid : /","RequestId":"5B55386A2CE41D1F4FBCF7E7"}}
```

更多检查方式请参考如何判断加速生效。

- 访问加速域名报错project not exist，应如何处理？

遇到此种情况一般是您配置了错误的源站地址，请在全站加速控制台将源站地址修改为您的Project所属Region的公网地址，地址列表参见：日志服务入口列表。

> 说明：
> 切换源站地址会有几分钟的同步延迟。

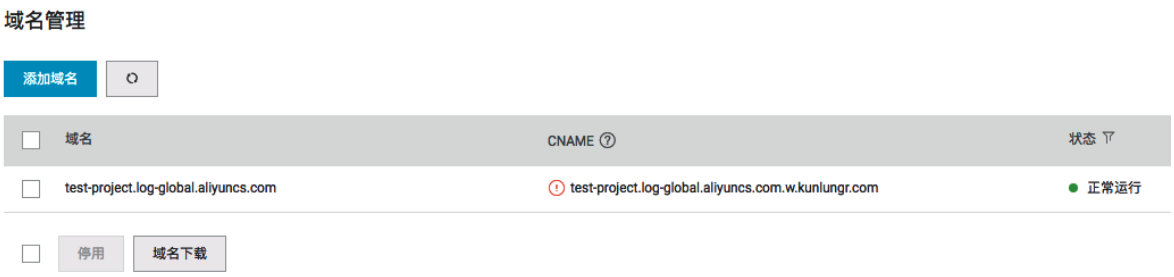
3.2.3 关闭全球加速

若您需要关闭日志服务全球加速功能，请按照以下方式进行关闭。

> 说明：
> 全球加速功能关闭后，开通时配置的加速域名将无法使用。请在关闭加速功能前，确保您所有客户端不会通过该域名上传或请求数据。

关闭全球加速功能

1. 登录全站加速控制台，单击左侧导航栏的域名管理，进入域名管理页面。
2. 查看需要关闭的域名对应的**CNAME**。



3. 登录日志服务控制台，在**Project**列表界面单击对应Project右侧的全球加速，进入全球加速页面。
4. 填入**CNAME**，并单击关闭加速。


全球加速
当前状态:已开启
* Project名称:etl-test-1
* 加速域名:etl-test-1.log-global.aliyuncs.com 复制
* CNAME: etl-test-1.log-global.aliyuncs.com.w.kunluncan.co
如何使用请参考： 全球加速使用说明
如何关闭请参考： 关闭全球加速
关闭加速
取消

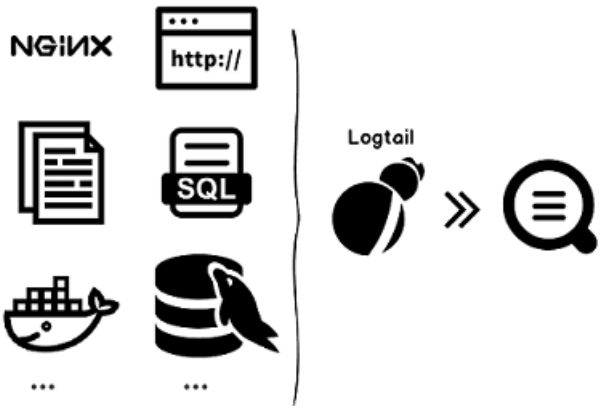
4 Logtail采集

4.1 简介

4.1.1 Logtail简介

Logtail接入服务是日志服务提供的日志采集Agent，通过控制台方式帮助您实时采集阿里云ECS、自建IDC、其他云厂商等服务器上的日志。

图 **4-1: Logtail采集功能**



功能优势

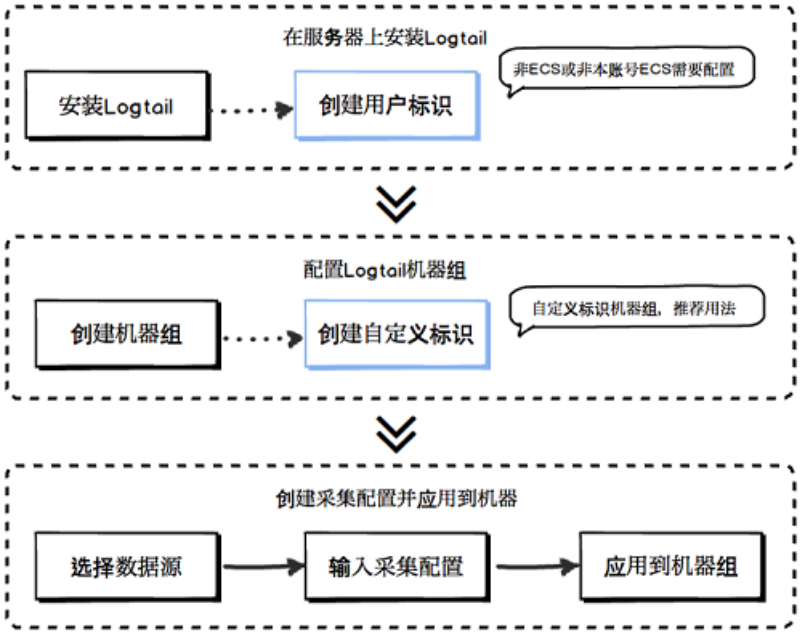
- 基于日志文件、无侵入式的收集日志。用户无需修改应用程序代码，且日志收集不会影响用户应用程序的运行逻辑。
- 除支持文本日志采集外，还支持binlog、http、容器stdout等采集方式。
- 对于容器支持友好，支持标准容器、swarm集群、Kubernetes集群等容器集群的数据采集。
- 能够稳定地处理日志收集过程中各种异常。当遇到网络异常、服务端异常等问题时会采用主动重试、本地缓存数据等措施保障数据安全。
- 基于服务端的集中管理能力。用户在安装Logtail后（参见 *Windows* 和 *Linux* ），只需要在服务端集中配置需要收集的机器、收集方式等信息即可，无需逐个登录服务器进行配置。
- 完善的自我保护机制。为保证运行在客户机器上的收集Agent不会明显影响用户自身服务的性能，Logtail客户端在CPU、内存及网络使用方面都做了严格的限制和保护机制。

处理能力与限制

参见相关限制说明。

配置流程

图 4-2: 配置流程



通过Logtail采集服务器日志可以通过以下步骤完成：

1. 安装Logtail。在需要采集日志的源服务器上安装Logtail操作请参见*Windows* 和 *Linux*。
2. 创建用户自定义标识机器组。从阿里云ECS采集日志不需要执行此步骤。
3. 创建*IP*地址机器组。日志服务通过机器组的方式管理所有需要通过Logtail客户端采集日志的服务器。日志服务支持通过IP或者自定义标识的方式定义机器组。您也可以在应用Logtail配置到机器组时，根据提示创建机器组。
4. 创建Logtail采集配置，并应用到机器组。您可以通过**数据接入向导**创建Logtail配置以 采集文本日志、*Syslog*等，并将该Logtail配置应用到机器组。

在完成如上流程后，您的ECS服务器上需要收集的新增日志会被主动收集、发送到对应Logstore中，历史数据不会被收集。您可以通过日志服务控制台或者SDK及API查询到这些日志。您还可以通过日志服务查询到所有ECS服务器上的Logtail收集日志状态，例如是否在正常收集，是否有错误等。

Logtail接入服务在日志服务控制台上的完整操作请参考*Logtail* 收集日志 。

容器

- 阿里云容器服务Swarm：参见集成日志服务。

- 阿里云容器服务Kubernetes：参见采集*Kubernetes*日志
- 自建Kubernetes：参见自建*Kubernetes*安装方式
- 自建其他Docker集群：参见标准*Docker*日志采集流程

核心概念

- **机器组**：一个机器组包含一或多台需要收集一类日志的机器。通过绑定Logtail配置到机器组，可以让日志服务根据同样的Logtail配置采集一个机器组内所有服务器上的日志。您也可以通过日志服务控制台方便地对机器组进行管理（包括创建、删除机器组，添加、移除机器等）。同一个机器组内不可同时包含Windows和Linux机器，但可以包含不同版本的Windows Server或者不同发行版本的Linux机器。
- **Logtail客户端**：Logtail是运行在需要收集日志的服务器上上执行日志收集工作的Agent。安装步骤请参考 *Windows* 和 *Linux*。 在服务器上安装Logtail后，需要配置Logtail并应用到机器组。
  - **Linux** 下，Logtail安装在 `/usr/local/ilogtail` 目录下，并启动两个以 ilogtail 开头的个独立进程，一个为收集进程，另外一个为守护进程，程序运行日志为 `/usr/local/ilogtail/ilogtail.LOG`。
  - **Windows** 下，Logtail安装在目录 `C:\Program Files\Alibaba\Logtail`（32 位系统）或 `C:\Program Files (x86)\Alibaba\Logtail`（64 位系统）下。您可以通过Windows管理工具>服务查看到两个Windows Service，LogtailWorker负责收集日志，LogtailDaemon负责守护工作程序。程序运行日志为安装目录下的 `logtail_*.log`。
- **Logtail配置**：是Logtail收集日志的策略集合。通过为Logtail配置数据源、收集模式等参数，来对机器组内所有服务器进行定制化的收集策略。Logtail配置定义了如何在机器上收集一类日志并解析、发送到日志服务的指定日志库。您可以通过控制台对每个Logstore添加Logtail配置，表示该Logstore接收以此Logtail配置收集的日志。

基本功能

Logtail接入服务提供如下功能：

- **实时收集日志**：动态监控日志文件，实时地读取、解析增量日志。日志从生成到发往服务端的延迟一般在3秒内。

  说明：

  Logtail接入服务不支持对历史数据的收集。对于一条日志，读取该日志的时刻减去日志产生的时刻，差值超过5分钟的会被丢弃。

- **自动处理日志轮转**：很多应用会按照文件大小或者日期对日志文件进行轮转（rotation），把原日志文件重命名，并新建一个空日志文件等待写入。例如：监控*app.LOG*，日志轮转会产生*app.LOG.1*，*app.LOG.2* 等。您可以指定收集日志写入的文件，如 *app.LOG*，Logtail会自动检测到日志轮转过程，保证这个过程中不会出现日志数据丢失。
- **多种采集输入源**：Logtail除支持文本日志采集外，还支持syslog、http、MySQL binlog等输入源，更多内容参见采集数据源配置章节。
- **兼容开源采集Agent**：Logtail支持Logstash、Beats等开源软件采集的数据作为输入源，更多内容参见采集数据源配置章节。
- **自动处理收集异常**：因为服务端错误、网络措施、Quota超限等各种异常导致数据发送失败，Logtail会按场景主动重试。如果重试失败则会将数据写入本地缓存，稍后自动重发。
- **灵活配置收集策略**：可以通过Logtail配置来非常灵活地指定如何在一台ECS服务器上收集日志。具体来说，您可以根据实际场景选择日志目录、文件，既可精确匹配，也可通过通配符模糊匹配。您可以自定义日志收集提取的方式和各个提取字段的名称，日志服务支持正则表达式方式的日志提取。另外，由于日志服务日志数据模型要求每条日志必须有精确的时间戳信息，Logtail提供了自定义的日志时间格式，方便您从不同格式的日志数据中提取必须要的日志时间戳信息。
- **自动同步收集配置**：您在日志服务控制台上新建或更新配置，Logtail一般在3分钟时间内即可自动接受并使之生效，更新配置过程中数据收集不丢失。
- **自动升级客户端**：在您手动安装Logtail到服务器后，日志服务负责Logtail 自动运维升级，此过程无需您参与。在整个Logtail升级过程中日志数据不丢失。
- **自我监控状态**：为避免Logtail客户端消耗您太多资源而影响您其他服务。Logtail客户端会实时监控自身CPU和内存消耗。如果Logtail客户端在运行过程中，资源使用超出限制将会自动重启，避免影响机器上的其它作业。同时，该客户端也会有主动的网络限流保护措施，防止过度消耗用户带宽。
- **签名数据发送**：为保证您的数据在发送过程中不会被篡改，Logtail客户端会主动获取用户的阿里云访问秘钥并对所有发送日志的数据包进行数据签名。



说明：

Logtail客户端在获取您的阿里云访问秘钥时采用HTTPS通道，保障您的访问秘钥安全性。

数据采集可靠性

Logtail在采集数据时，会定期将采集的点位（CheckPoint）信息保存到本地，若遇到服务器意外关闭、进程崩溃等异常情况时，Logtail重启后会从上一次记录的位置处开始采集数据，尽可能保

证数据不丢失。Logtail会根据配置文件中的资源限制进行工作，若资源占用超过限定值5分钟以上，则Logtail会强制重启。重启后可能会产生一定的数据重复。

Logtail内部采用了很多机制提升日志采集可靠性，但并不能保证日志绝对不会丢失。以下情况可能造成日志丢失：

- Logtail未运行且日志轮转多次。
- 日志轮转速度极快，例如1秒轮转1次。
- 日志采集速度长期无法达到日志产生速度。

4.1.2 Logtail采集原理

使用Logtail客户端采集服务器日志时，Logtail采集日志的流程为：监听文件、读取文件、处理日志、过滤日志、聚合日志和发送数据6个步骤。

在服务器上安装Logtail客户端，并添加Logtail采集配置之后，Logtail即时开始采集日志到日志服务。Logtail的日志采集流程如下：

1. 监听文件
2. 读取文件
3. 处理日志
4. 过滤日志
5. 聚合日志
6. 发送日志



说明：

将Logtail采集配置应用到机器组之后，机器组中服务器上没有发生修改事件的日志文件会被判定为历史文件。Logtail在正常运行模式中不支持采集历史文件，若您需要采集历史日志，请参考导入历史日志文件。

监听文件

在服务器上安装Logtail客户端，并根据数据源添加Logtail采集配置之后，Logtail采集配置从服务端实时下发到Logtail。Logtail根据采集配置开始监听文件。

1. Logtail根据配置的日志路径和最大监控目录深度逐层扫描目录下符合指定文件名规则的日志目录和文件。

为保证日志采集时效性以及稳定性，Logtail会对采集目录注册事件监听（Linux下*Inotify*、Windows下使用*ReadDirectoryChangesW*）以及定期轮询。

2. Logtail如果监听到指定目录下符合规则的日志文件在应用配置之后没有修改过，则不会采集；如果有日志文件产生了修改事件，会触发采集流程，Logtail开始读取文件。

读取文件

确定日志文件有更新后，Logtail开始读取文件。

1. 若该文件首次读取，会检查文件大小。
   - 如果文件小于1 MB，则从文件内容起始位置开启读取。
   - 如果文件大于1 MB，则从文件末尾1 MB处开始读取。
2. 如果该文件曾被Logtail读取过，则从上次读取的Checkpoint处继续读取。
3. 读取文件时，每次最多可以读取512KB，因此每条日志请控制在512KB以内，否则无法正常读取。

处理日志

logtail读取日志后，对日志内容进行分行、解析，并确认日志时间字段。

1. 分行：

   如果Logtail采集配置中指定了**行首正则**，则根据行首配置对Logtail每次读取的日志数据块进行分行，切分成多条日志；如果没有指定，则将一个数据块作为一条日志处理。

2. 解析：

   根据Logtail采集配置，对每条日志内容执行对应的解析，例如正则、分隔符、JSON等。

   > 说明：
   > 如果您的正则式较为复杂，可能会导致CPU占用率过高，请使用合理高效的正则表达式。

3. 解析失败处理：

   根据Logtail采集配置中是否开启丢弃解析失败日志功能，判断日志解析失败的处理方式。

   - 开启**丢弃解析失败日志**，则直接丢弃该日志，并上报解析失败的报错信息。
   - 关闭 **丢弃解析失败日志**，则上传解析失败的原始日志，其中Key为**raw_log**、Value为日志内容。

4. 设置日志时间字段：

- 若未配置时间字段，则日志时间为当前解析时间。
- 若配置了时间字段：
  - 日志记录的时间距离当前时间12小时以内，则从解析的日志字段中提取时间。
  - 日志记录的时间距离当前时间12小时以上，则丢弃该日志并上传错误信息。

过滤日志

处理日志后，根据Logtail采集配置中的过滤器配置过滤日志。

- 未设置**过滤器配置**：不过滤日志，执行下一个步骤。
- 已设置**过滤器配置**：对每条日志中的所有字段进行遍历并验证。
  - 符合过滤器配置的日志：如果日志中出现了过滤器中配置的所有字段、且所有对应的字段全部符合配置，则采集该条日志。
  - 不符合过滤器配置的日志：不符合过滤器配置的日志不会被采集。

聚合日志

通过过滤器配置过滤日志后，符合配置的日志数据将发往日志服务。为降低网络请求次数，当日志处理、过滤完毕后，会在Logtail内部缓存一段时间，进行聚合打包，再发送到日志服务。

缓存过程中，如果满足以下条件之一，日志将即时打包发送到日志服务。

- 日志聚合时间超过3秒。
- 日志聚合条数超过4096条。
- 日志聚合总大小超过512 KB。

发送日志

Logtail将采集到的日志数据聚合发送到日志服务。设置启动参数配置中的参数`max_bytes_per_sec`和`send_request_concurrency`可以调整日志数据的发送速度和最大并发数，Logtail会保证发送速率以及并发不超过配置值。

若数据发送失败，Logtail自动根据错误信息决定重试或放弃发送。

错误信息	说明	Logtail处理方式
401错误	Logtail客户端没有权限采集数据。	直接丢弃日志包。
404错误	Logtail采集配置中指定的Project或Logstore不存在。	直接丢弃日志包。

错误信息	说明	Logtail处理方式
403错误	Shard Quota超出限制。	等待3秒后重试。
500错误	服务端异常。	等待3秒后重试。
网络超时	网络连接错误。	等待3秒后重试。

4.1.3 Logtail配置和记录文件

Logtail运行时会依赖一系列的配置文件并产生部分信息记录文件，本文档介绍常见文件的基本信息及路径。

配置文件：

- 启动配置文件*#ilogtail_config.json#*
- *AliUid*配置文件
- 用户自定义标识文件*#user_defined_id#*
- 采集配置文件*#user_log_config.json#*

记录文件：

- *AppInfo*记录文件*#app_info.json#*
- *Logtail*运行日志*#ilogtail.LOG#*
- *Logtail*插件日志*#logtail_plugin.LOG#*
- 容器路径映射文件*#docker_path_config.json#*

启动配置文件（ilogtail_config.json）

启动配置文件（ilogtail_config.json）用来查看或配置Logtail的运行参数，文件类型为JSON。

安装Logtail后，您可以通过该文件：

- 修改Logtail的运行参数。

  可以通过修改启动配置文件来修改CPU使用阈值、常驻内存使用阈值等配置信息。
- 检验安装命令是否正确。

  该文件中的`config_server_address`和`data_server_list`取决于安装时选择的参数和安装命令，如果其中的区域和日志服务所在区域不一致或地址无法联通，说明安装时选择了错误的参数或命令。这时Logtail无法正常采集日志，需要重新安装。



说明：

- 该文件必须为合法JSON，否则无法启动Logtail。
- 修改该文件后需重启Logtail才能生效。

默认配置项如下，您也可以参考配置启动参数写入其他配置。

表 4-1: 启动配置文件默认配置项

配置项	说明
config_server_address	Logtail从服务端获取配置文件的地址，取决于安装时选择的参数和安装命令。 请保证该地址能够联通，且其中的区域和日志服务所在区域一致。
data_server_list	数据服务器地址，取决于安装时选择的参数和安装命令。 请保证该地址能够联通，且其中的区域和日志服务所在区域一致。
cluster	区域名称。
endpoint	服务入口。
cpu_usage_limit	CPU 使用阈值，以单核计算。
mem_usage_limit	常驻内存使用阈值。
max_bytes_per_sec	Logtail 发送原始数据的流量限制，超过20 MB/s则不限流。
process_thread_count	Logtail 处理日志文件写入数据的线程数。
send_request_concurrency	异步并发的个数。Logtail 默认异步发送数据包，如果写入TPS很高，可以配置更高的异步并发。

文件地址：

- Linux：*/usr/local/ilogtail/ilogtail_config.json*。
- 容器：该文件存储在Logtail容器中，文件地址配置在Logtail容器的环境变量`ALIYUN_LOGTAIL_CONFIG`中，可通过`docker inspect ${logtail_container_name} | grep ALIYUN_LOGTAIL_CONFIG`查看，例如*/etc/ilogtail/conf/cn-hangzhou/ilogtail_config.json*。
- Windows：
  - x64：*C:\Program Files (x86)\Alibaba\Logtail\ilogtail_config.json*。

- x32：*C:\Program Files\Alibaba\Logtail\ilogtail_config.json*。

文件示例：

```
$cat /usr/local/ilogtail/ilogtail_config.json
{
    "config_server_address" : "http://logtail.cn-hangzhou-intranet.log
.aliyuncs.com",
    "data_server_list" :
    [
        {
            "cluster" : "ap-southeast-2",
            "endpoint" : "cn-hangzhou-intranet.log.aliyuncs.com"
        }
    ],
    "cpu_usage_limit" : 0.4,
    "mem_usage_limit" : 100,
    "max_bytes_per_sec" : 2097152,
    "process_thread_count" : 1,
    "send_request_concurrency" : 4,
    "streamlog_open" : false
}
```

AliUid配置文件

AliUid配置文件中包含阿里云账号的AliUid账号信息，主要用于标识这台服务器有权限被该账号访问、采集日志。采集非本账号ECS、自建IDC的日志时，需要手动创建AliUid配置文件。详细说明以及配置参见为非本账号*ECS*、自建*IDC*配置*AliUid*。



说明：

- 该文件为可选配置，仅在采集非本账号ECS、自建IDC日志时使用。
- AliUid文件必须为主账号AliUid，不支持子账号。
- AliUid文件只需配置文件名即可，文件不能有后缀。
- 一个Logtail可配置多个AliUid文件，Logtail容器仅可配置一个AliUid文件。

文件地址

- Linux：`/etc/ilogtail/users/`。
- 容器：该AliUid直接配置在Logtail容器的环境变量`ALIYUN_LOGTAIL_USER_ID`中，可通过`docker inspect ${logtail_container_name} | grep ALIYUN_LOGTAIL_USER_ID`查看。
- Windows：`C:\LogtailData\users\`。

文件示例

```
$ls /etc/ilogtail/users/
```

```
1559122535028493 1329232535020452
```

用户自定义标识文件（user_defined_id）

用户自定义标识文件（user_defined_id）用于配置自定义标识机器组，详细说明以及配置参见创建用户自定义标识机器组。



说明：

- 该文件为可选配置，只有在配置自定义标识机器组时使用。
- 若配置多个自定义标识，使用换行符分隔。

文件地址

- Linux：*/etc/ilogtail/user_defined_id*。
- 容器：该标识直接配置在Logtail容器的环境变量`ALIYUN_LOGTAIL_USER_DEFINED_ID`中，可通过*docker inspect ${logtail_container_name} | grep ALIYUN_LOGTAIL_USER_DEFINED_ID*查看。
- Windows：*C:\LogtailData\user_defined_id*。

文件示例

```
$cat /etc/ilogtail/user_defined_id
aliyun-ecs-rs1e16355
```

采集配置文件（user_log_config.json）

该文件记录Logtail从服务端获取的采集配置信息，文件类型为JSON，每次配置更新时会同步更新该文件。可通过该文件确认Logtail配置是否已经下发到该服务器。采集配置文件存在，且内容为最新，表示Logtail配置已下发。



说明：

- 除手动配置密钥信息、数据库密码等敏感信息外，不建议修改该文件。
- 提交工单时，请上传此文件。

文件地址

- Linux：*/usr/local/ilogtail/user_log_config.json*。
- 容器：*/usr/local/ilogtail/user_log_config.json*。
- Windows

- x64：*C:\Program Files (x86)\Alibaba\Logtail\user_log_config.json*。
- x32：*C:\Program Files\Alibaba\Logtail\user_log_config.json*。

文件示例

```
$cat /usr/local/ilogtail/user_log_config.json
{
   "metrics" : {
      "##1.0##k8s-log-c12ba2028*****939f0b$app-java" : {
         "aliuid" : "16542189*****50",
         "category" : "app-java",
         "create_time" : 1534739165,
         "defaultEndpoint" : "cn-hangzhou-intranet.log.aliyuncs.com",
         "delay_alarm_bytes" : 0,
         "enable" : true,
         "enable_tag" : true,
         "filter_keys" : [],
         "filter_regs" : [],
         "group_topic" : "",
         "local_storage" : true,
         "log_type" : "plugin",
         "log_tz" : "",
         "max_send_rate" : -1,
         "merge_type" : "topic",
         "plugin" : {
            "inputs" : [
               {
                  "detail" : {
                     "IncludeEnv" : {
                        "aliyun_logs_app-java" : "stdout"
                     },
                     "IncludeLable" : {
                        "io.kubernetes.container.name" : "java-log-
demo-2",
                        "io.kubernetes.pod.namespace" : "default"
                     },
                     "Stderr" : true,
                     "Stdout" : true
                  },
                  "type" : "service_docker_stdout"
               }
            ]
         },
         "priority" : 0,
         "project_name" : "k8s-log-c12ba2028c*****ac1286939f0b",
         "raw_log" : false,
         "region" : "cn-hangzhou",
         "send_rate_expire" : 0,
         "sensitive_keys" : [],
         "tz_adjust" : false,
         "version" : 1
      }
   }
```

```
}
```

AppInfo记录文件（app_info.json）

AppInfo记录文件（app_info.json）记录Logtail的启动时间、获取到的IP地址、hostname等信息。配置*IP*地址机器组时，需要在该文件中查看Logtail获取到的IP地址。

通常情况下，Logtail根据以下规则获取服务器IP地址：

- 如果已在服务器文件*/etc/hosts*中设置了主机名与IP地址绑定，则自动获取绑定的IP地址。
- 如果没有设置主机名绑定，会自动获取本机的第一块网卡的IP地址。



说明：

- AppInfo记录文件仅用于记录Logtail内部信息，手动修改文件内容不能改变Logtail的基本信息。
- 若修改了服务器的hostname等网络配置，请重新启动Logtail以获取新的IP地址。

表 4-2: 字段说明

字段	说明
UUID	服务器序列号。
hostname	主机名。
instance_id	随机生成的Logtail唯一标识。
ip	Logtail获取到的IP地址。该字段为空时表示Logtail没有获取到IP地址，Logtail无法正常运行。请为服务器设置IP地址并重启Logtail。  说明： 如果机器组为IP地址机器组，请确保机器组中配置的IP与此处显示的IP地址一致。若服务端机器组填写了错误的IP地址，请修改机器组内IP地址并保存，等待1分钟再查看。
logtail_version	Logtail客户端版本。
os	操作系统版本。
update_time	Logtail最近一次启动时间。

文件地址

- Linux：*/usr/local/ilogtail/app_info.json*。

- 容器：*/usr/local/ilogtail/app_info.json*。
- Windows
  - x64：*C:\Program Files (x86)\Alibaba\Logtail\app_info.json*。
  - x32：*C:\Program Files\Alibaba\Logtail\app_info.json*。

文件示例

```
$cat /usr/local/ilogtail/app_info.json
{
   "UUID" : "",
   "hostname" : "logtail-ds-slpn8",
   "instance_id" : "E5F93BC6-B024-11E8-8831-0A58AC14039E_172.20.3.
158_1536053315",
   "ip" : "172.20.3.158",
   "logtail_version" : "0.16.13",
   "os" : "Linux; 3.10.0-693.2.2.el7.x86_64; #1 SMP Tue Sep 12 22:26:
13 UTC 2017; x86_64",
   "update_time" : "2018-09-04 09:28:36"
}
```

Logtail运行日志（ilogtail.LOG）

Logtail运行日志（ilogtail.LOG）记录Logtail客户端的运行信息，日志级别从低到高分别为INFO、WARN和ERROR，其中INFO类型日志无需关注。

 说明：

- 请先诊断采集错误，根据具体的错误类型和Logtail运行日志排查问题。
- 若因Logtail采集异常提交工单时，请同时上传该日志。

文件地址

- Linux：*/usr/local/ilogtail/ilogtail.LOG*。
- 容器：*/usr/local/ilogtail/ilogtail.LOG*。
- Windows
  - x64：*C:\Program Files (x86)\Alibaba\Logtail\logtail_*.log*。
  - x32：*C:\Program Files\Alibaba\Logtail\logtail_*.log*。

文件示例

```
$tail /usr/local/ilogtail/ilogtail.LOG
[2018-09-13 01:13:59.024679]    [INFO]    [3155]    [build/release64
/sls/ilogtail/elogtail.cpp:123]    change working dir:/usr/local/
ilogtail/
```

```
[2018-09-13 01:13:59.025443]    [INFO]    [3155]    [build/release64/
sls/ilogtail/AppConfig.cpp:175]    load logtail config file, path:/etc
/ilogtail/conf/ap-southeast-2/ilogtail_config.json
[2018-09-13 01:13:59.025460]    [INFO]    [3155]    [build/release64/
sls/ilogtail/AppConfig.cpp:176]    load logtail config file, detail:{
   "config_server_address" : "http://logtail.ap-southeast-2-intranet.
log.aliyuncs.com",
   "data_server_list" : [
      {
         "cluster" : "ap-southeast-2",
         "endpoint" : "ap-southeast-2-intranet.log.aliyuncs.com"
      }
]
```

Logtail插件日志（logtail_plugin.LOG）

Logtail插件日志（logtail_plugin.LOG）记录容器标准输出、binlog、http等插件的运行信息，日志级别从低到高分别为`INFO`、`WARN`和`ERROR`，其中`INFO`类型日志无需关注。

诊断采集错误时，若有**CANAL_RUNTIME_ALARM**等插件错误，可以根据Logtail插件日志进行排查。

 说明：

若因插件异常提交工单时，请同时上传该日志。

文件地址

- Linux：*/usr/local/ilogtail/logtail_plugin.LOG*。
- 容器：*/usr/local/ilogtail/logtail_plugin.LOG*。
- Windows：不支持插件功能。

文件示例

```
$tail /usr/local/ilogtail/logtail_plugin.LOG
2018-09-13 02:55:30 [INF] [docker_center.go:525] [func1] docker fetch
all:start
2018-09-13 02:55:30 [INF] [docker_center.go:529] [func1] docker fetch
all:stop
2018-09-13 03:00:30 [INF] [docker_center.go:525] [func1] docker fetch
all:start
2018-09-13 03:00:30 [INF] [docker_center.go:529] [func1] docker fetch
all:stop
2018-09-13 03:03:26 [INF] [log_file_reader.go:221] [ReadOpen] [##1.
0##sls-zc-test-hz-pub$docker-stdout-config,k8s-stdout]    open file
 for read, file:/logtail_host/var/lib/docker/containers/7f46afec6a
14de39b59ee9cdfbfa8a70c2fa26f1148b2e2f31bd3410f5b2d624/7f46afec6a
14de39b59ee9cdfbfa8a70c2fa26f1148b2e2f31bd3410f5b2d624-json.log
offset:40379573    status:794354-64769-40379963
2018-09-13 03:03:26 [INF] [log_file_reader.go:221] [ReadOpen] [##1
.0##k8s-log-c12ba2028cfb444238cd9ac1286939f0b$docker-stdout-config
,k8s-stdout]    open file for read, file:/logtail_host/var/lib/
docker/containers/7f46afec6a14de39b59ee9cdfbfa8a70c2fa26f1148b2e2f31
```

```
bd3410f5b2d624/7f46afec6a14de39b59ee9cdfbfa8a70c2fa26f1148b2e2f31
bd3410f5b2d624-json.log    offset:40379573    status:794354-64769-
40379963
2018-09-13 03:04:26 [INF] [log_file_reader.go:308] [CloseFile] [##1.0
##sls-zc-test-hz-pub$docker-stdout-config,k8s-stdout]    close file,
reason:no read timeout    file:/logtail_host/var/lib/docker/containers
/7f46afec6a14de39b59ee9cdfbfa8a70c2fa26f1148b2e2f31bd3410f5b2d624/
7f46afec6a14de39b59ee9cdfbfa8a70c2fa26f1148b2e2f31bd3410f5b2d624-json.
log    offset:40379963    status:794354-64769-40379963
2018-09-13 03:04:27 [INF] [log_file_reader.go:308] [CloseFile] [##1.
0##k8s-log-c12ba2028cfb444238cd9ac1286939f0b$docker-stdout-config,k8s
-stdout]    close file, reason:no read timeout    file:/logtail_host
/var/lib/docker/containers/7f46afec6a14de39b59ee9cdfbfa8a70c2fa26f1
148b2e2f31bd3410f5b2d624/7f46afec6a14de39b59ee9cdfbfa8a70c2fa26f1
148b2e2f31bd3410f5b2d624-json.log    offset:40379963    status:794354-
64769-40379963
2018-09-13 03:05:30 [INF] [docker_center.go:525] [func1] docker fetch
all:start
2018-09-13 03:05:30 [INF] [docker_center.go:529] [func1] docker fetch
all:stop
```

容器路径映射文件（**docker_path_config.json**）

容器路径映射文件（docker_path_config.json）只有在采集容器文件时才会自动创建，用于记录容器文件和实际文件的路径映射关系。文件类型为JSON。

诊断采集错误时，如果报错信息为**DOCKER_FILE_MAPPING_ALARM**，表示执行Logtail命令添加Docker文件映射失败，可以通过容器路径映射文件排查问题。



说明：

- 该文件为信息记录文件，任何修改操作均不会生效；删除后会自动创建，不影响业务的正常运行。
- 因容器日志采集异常而提交工单时，请同时在工单中上传此文件。

文件地址

/usr/local/ilogtail/docker_path_config.json。

文件示例

```
$cat /usr/local/ilogtail/docker_path_config.json
{
   "detail" : [
      {
         "config_name" : "##1.0##k8s-log-c12ba2028cfb444238cd
9ac1286939f0b$nginx",
         "container_id" : "df19c06e854a0725ea7fca7e0378b0450f7bd312
2f94fe3e754d8483fd330d10",
         "params" : "{\n   \"ID\" : \"df19c06e854a0725ea7fca7e0378b0
450f7bd3122f94fe3e754d8483fd330d10\",\n   \"Path\" : \"/logtail_ho
st/var/lib/docker/overlay2/947db346695a1f65e63e582ecfd10ae1f57019a1
b99260b6c83d00fcd1892874/diff/var/log\",\n   \"Tags\" : [\n      \"
nginx-type\",\n      \"access-log\",\n      \"_image_name_\",\n
```

```
   \"registry.cn-hangzhou.aliyuncs.com/log-service/docker-log-test
:latest\",\n      \"_container_name_\",\n      \"nginx-log-demo\",
\n      \"_pod_name_\",\n      \"nginx-log-demo-h2lzc\",\n      \"
_namespace_\",\n      \"default\",\n      \"_pod_uid_\",\n      \"
87e56ac3-b65b-11e8-b172-00163f008685\",\n      \"_container_ip_\",\n
    \"172.20.4.224\",\n      \"purpose\",\n      \"test\"\n   ]\n}\n"
      }
   ],
   "version" : "0.1.0"
}
```

4.2 选择网络

采集日志数据到日志服务时，日志数据可以通过阿里云内网、公网和全球加速网络传输。

网络类型

- 公网：使用公网传输日志数据，不仅会受到网络带宽的限制，还可能会因网络抖动、延迟、丢包等影响数据采集的速度和稳定性。
- 阿里云内网：阿里云内网为千兆共享网络，日志数据通过阿里云内网传输比公网传输更快速、稳定。内网包括**VPC**环境和经典网络环境。
- 全球加速：利用阿里云CDN边缘节点进行日志采集加速，相对公网采集在网络延迟、稳定性上具有很大优势。

如何选择网络

- 内网：

  您的日志数据是否通过阿里云内网传输，取决于您的服务器类型以及服务器和日志服务Project是否在同一地域。仅有以下两种情况可以使用阿里云内网传输：

  - 本账号下的**ECS**和日志服务**Project**在同一地域。
  - 其他账号的**ECS**和本账号的日志服务**Project**在同一地域。

  因此，建议您在ECS的相同地域下创建日志服务Project，并将日志采集到同地域的日志服务Project中。ECS上的日志数据自动通过阿里云内网写入日志服务，不消耗公网带宽。

  > 说明：
  > 在服务器上安装Logtail时，选择的地域必须和Project所在地域一致，否则无法正常采集日志数据。

- 全球加速：

如果您的服务器分布在海外各地的自建机房、或者来自海外云厂商，使用公网传输数据可能会出现网络延迟高、传输不稳定等问题，可以通过全球加速传输数据。全球加速利用阿里云CDN边缘节点进行日志采集加速，相对公网采集在网络延迟、稳定性上具有很大优势。

- 公网：

  在以下两种情况时，您可以选择网络类型为公网：

  - 服务器为ECS，但和日志服务Project位于不同地域。
  - 服务器为其他云厂商服务器、自建IDC。

服务器类型	是否与Project同一地域	是否需要配置AliUid	网络类型
本账号下的ECS	同一地域	不需要	阿里云内网
	不同地域	不需要	公网或全球加速
其他账号下的ECS	同一地域	需要	阿里云内网
	不同地域	需要	公网或全球加速
其他云厂商服务器、自建IDC	-	需要	公网或全球加速

说明：

日志服务无法获取非本账号下ECS、其他服务器的属主信息，请在安装Logtail后手动配置用户标识（AliUid），否则安装Logtail的服务器会心跳异常、无法收集日志。详细步骤请参见 为非本账号*ECS*、自建*IDC*配置*AliUid*。

网络选择示例

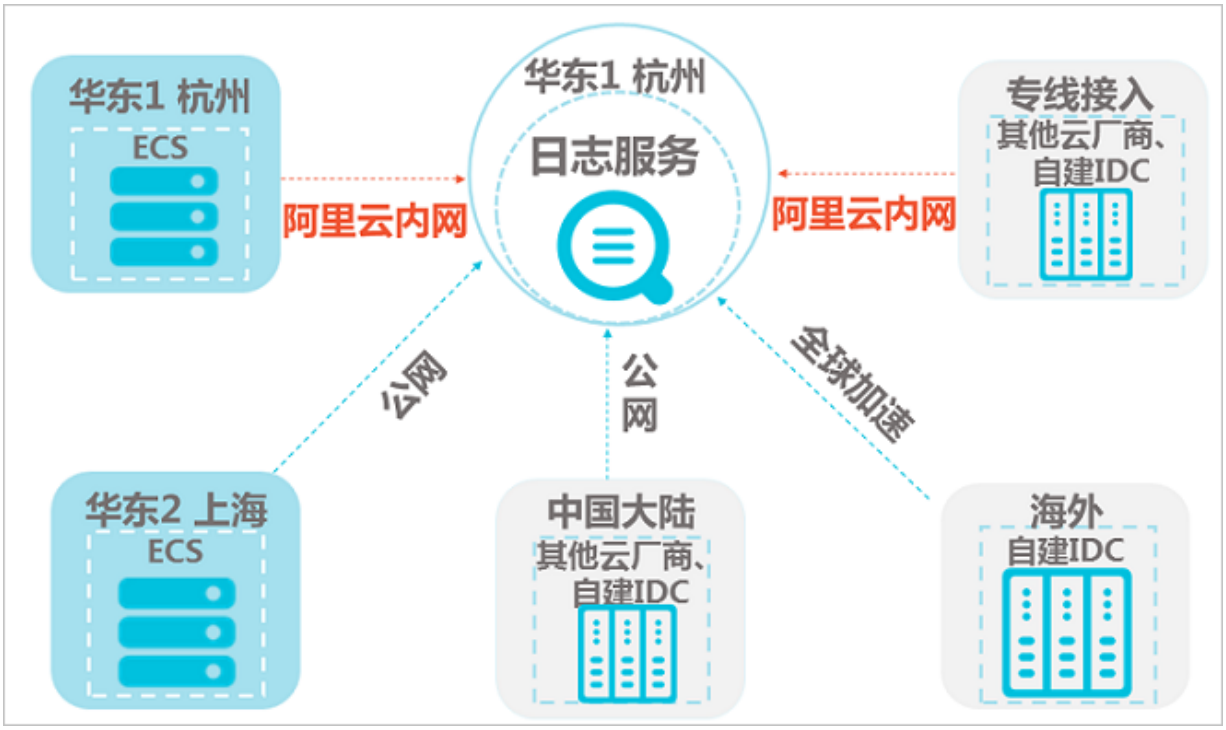
以下是各种常见场景的网络选择示例，请根据您的实际场景选择网络类型。

说明：

全球加速场景中，日志服务Project创建在香港地域，服务器为全球各地的自建机房，数据采集的速度和可靠性尤为重要，所以建议您在类似场景下安装Logtail时选择香港地域的全球加速网络类型。日志数据通过全球加速传输，比公网传输的网络稳定性更高、性能更好。

场景类型	日志服务Project地域	服务器类型	ECS地域	安装Logtail时选择的地域	网络类型	是否需要配置AliUid
相同地域场景	华东1（杭州）	本账号ECS	华东1（杭州）	华东1（杭州）	内网	不需要
不同地域场景	华东2（上海）	本账号ECS	华北1（北京）	华北1（北京）	公网	不需要
其他账号场景	华东2（上海）	其他账号ECS	华北1（北京）	华北1（北京）	公网	需要
本地机房场景	华东5（深圳）	自建IDC	-	华东5（深圳）	公网	需要
全球加速场景	香港	自建IDC	-	香港	全球加速	需要

图 **4-3:** 网络选择示例



经典网络切换VPC后的配置更新

安装Logtail后，如果您的ECS网络类型由经典网络切换为VPC，请参考以下步骤更新配置。

1. 以管理员身份重启Logtail。

- **Linux**：

```
sudo /etc/init.d/ilogtaild stop
sudo /etc/init.d/ilogtaild start
```

- **Windows**：

  打开控制面板中的管理工具，打开服务，找到*LogtailWorker*并右键单击重新启动。

2. 更新机器组配置。

   - 自定义标识

     若机器组中配置了自定义标识，则无需手动更新机器组配置，可以直接正常使用VPC网络。

   - **IP地址**

     若机器组中配置了ECS云服务器IP地址，则需将机器组内的IP更换为重启Logtail后获取的IP地址，即*app_info.json*中的ip字段。

     *app_info.json*文件地址：

     - Linux：*/usr/local/ilogtail/app_info.json*
     - Windows x64：*C:\Program Files (x86)\Alibaba\Logtail\app_info.json*
     - Windows x32：*C:\Program Files\Alibaba\Logtail\app_info.json*

4.3 安装

4.3.1 Linux

Logtail客户端是日志服务提供的日志采集客户端，请参考本文档，在Linux服务器上安装Logtail客户端。

支持的系统

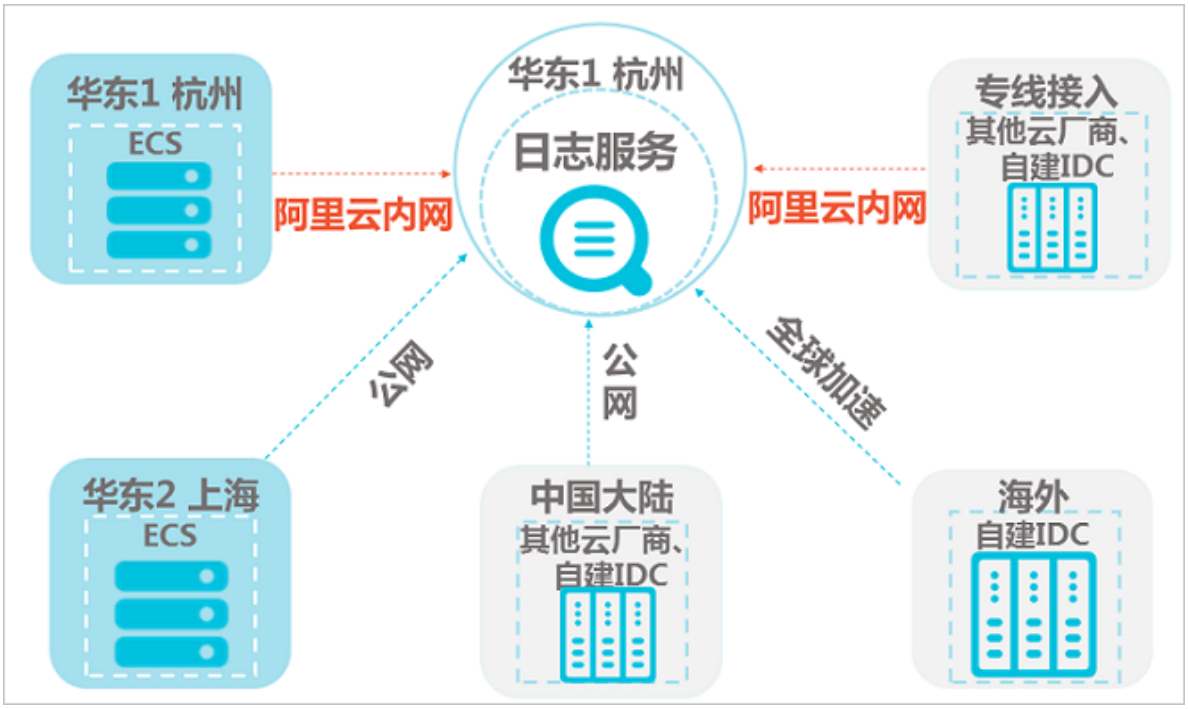
支持如下版本的Linux x86-64（64位）服务器：

- Aliyun Linux
- Ubuntu
- Debian
- CentOS
- OpenSUSE

前提条件

1. 已拥有一台及以上的服务器。
2. 已根据服务器类型和所在Region，确定日志采集流量的网络类型。详细说明请参考选择网络。

图 4-4: 选择网络



注意事项

- Logtail采用覆盖安装模式，若您之前已安装过Logtail，那么安装器会先执行卸载、删除`/usr/local/ilogtail`目录后再重新安装。安装后默认启动Logtail并注册开机启动。
- Docker和Kubernetes安装中的`${your_region_name}`即为安装参数中的参数，请直接拷贝。
- 如果安装失败，单击工单系统提交工单。

安装方式

选择网络后，请根据您的网络类型选择对应的安装命令。

- 阿里云内网#经典网络、*VPC#*
- 公网
- 全球加速
- *ECS* 金融云

执行安装命令之前，您需要将安装命令中的*${your_region_name}*替换为您的区域名称。各区域的安装参数如下，您也可以直接拷贝执行对应区域和网络类型的安装命令。

表 4-3: Logtail安装参数

区域	安装参数	区域	安装参数
华东 1（杭州）	cn-hangzhou	亚太东南 2（悉尼）	ap-southeast-2
华东 2（上海）	cn-shanghai	亚太东南 3（吉隆坡）	ap-southeast-3
华北 1（青岛）	cn-qingdao	亚太东南 5（雅加达）	ap-southeast-5
华北 2（北京）	cn-beijing	亚太南部 1（孟买）	ap-south-1
华北 3（张家口）	cn-zhangjiakou	亚太东北 1（日本）	ap-northeast-1
华北 5（呼和浩特）	cn-huhehaote	欧洲中部 1（法兰克福）	eu-central-1
华南 1（深圳）	cn-shenzhen	中东东部 1（迪拜）	me-east-1
西南 1（成都）	cn-chengdu	英国（伦敦）	eu-west-1
香港	cn-hongkong	华东 **1**金融云（杭州）	cn-hangzhou-finance
美国西部 1（硅谷）	us-west-1	华东 **2**金融云（上海）	cn-shanghai-finance
美国东部 1（弗吉尼亚）	us-east-1	华南 **1**金融云（深圳）	cn-shenzhen-finance
亚太东南 1（新加坡）	ap-southeast-1	-	-

阿里云内网（经典网络、VPC）

阿里云内网为千兆共享网络，日志数据通过阿里云内网传输比公网传输更快速、稳定，且不消耗公网带宽。

可以使用阿里云内网的场景：

- 服务器为阿里云ECS。
- ECS和日志服务Project位于同一区域。

执行安装命令时，需要根据区域选择安装参数，您可以选择**自动选择安装参数**和**手动安装**两种方式。

- **自动选择安装参数**

如果您无法确定ECS所在的区域，可以使用Logtail安装器的auto参数进行安装，当指定该参数后，Logtail 安装器会通过服务器获取您的实例元数据，自动确定ECS所在区域。

1. 通过公网下载Logtail 安装器。该步骤涉及访问公网，会消耗公网流量，约10KB左右。

```
wget http://logtail-release-cn-hangzhou.oss-cn-hangzhou.aliyuncs.
com/linux64/logtail.sh -O logtail.sh;chmod 755 logtail.sh
```

2. 使用auto参数进行安装。该步骤会自动下载对应区域的安装程序，不消耗公网流量。

```
./logtail.sh install auto
```

- 手动安装

您也可以选择手动安装Logtail。通过内网下载Logtail安装器，不消耗公网流量。

1. 根据日志服务**Project**所在区域选择安装参数。

安装命令中的`${your_region_name}`表示日志服务Project所在区域，根据安装参数选择正确参数，如华东一区域的安装参数为`cn-hangzhou`。

2. 替换参数后执行安装命令。

替换参数`${your_region_name}`后，执行安装命令。

```
wget http://logtail-
release-${your_region_name}.oss-${your_region_name}-
internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755
 logtail.sh; ./logtail.sh install ${your_region_name}
```

您也可以根据日志服务**Project**所在的区域直接执行下述对应的命令进行安装：

日志服务**Project**所在的区域	安装命令
华东 1（杭州）	`wget http://logtail-release-cn-hangzhou.oss-cn-hangzhou-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-hangzhou`
华东 2（上海）	`wget http://logtail-release-cn-shanghai.oss-cn-shanghai-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-shanghai`
华北 1（青岛）	`wget http://logtail-release-cn-qingdao.oss-cn-qingdao-internal.aliyuncs.com/linux64/logtail.sh`

日志服务Project所在的区域	安装命令
	-O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-qingdao
华北 2（北京）	wget http://logtail-release-cn-beijing.oss-cn-beijing-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-beijing
华北 3（张家口）	wget http://logtail-release-cn-zhangjiakou.oss-cn-zhangjiakou-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-zhangjiakou
华北 5（呼和浩特）	wget http://logtail-release-cn-huhehaote.oss-cn-huhehaote-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-huhehaote
华南 1（深圳）	wget http://logtail-release-cn-shenzhen.oss-cn-shenzhen-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-shenzhen
西南 1（成都）	wget http://logtail-release-cn-chengdu.oss-cn-chengdu-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-chengdu
香港	wget http://logtail-release-cn-hongkong.oss-cn-hongkong-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-hongkong
美国西部 1（硅谷）	wget http://logtail-release-us-west-1.oss-us-west-1-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install us-west-1
美国东部 1（弗吉尼亚）	wget http://logtail-release-us-east-1.oss-us-east-1-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install us-east-1
亚太东南 1（新加坡）	wget http://logtail-release-ap-southeast-1.oss-ap-southeast-1-internal.aliyuncs.com/linux64/

日志服务Project所在的区域	安装命令
	logtail.sh -O logtail.sh; chmod 755 logtail.sh ; ./logtail.sh install ap-southeast-1
亚太东南 2（悉尼）	wget http://logtail-release-ap-southeast-2.oss-ap-southeast-2-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh ; ./logtail.sh install ap-southeast-2
亚太东南 3（吉隆坡）	wget http://logtail-release-ap-southeast-3.oss-ap-southeast-3-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh ; ./logtail.sh install ap-southeast-3
亚太东南 5（雅加达）	wget http://logtail-release-ap-southeast-5.oss-ap-southeast-5-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh ; ./logtail.sh install ap-southeast-5
亚太东北 1（日本）	wget http://logtail-release-ap-northeast-1.oss-ap-northeast-1-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh ; ./logtail.sh install ap-northeast-1
亚太南部 1（孟买）	wget http://logtail-release-ap-south-1.oss-ap-south-1-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install ap-south-1
欧洲中部 1（法兰克福）	wget http://logtail-release-eu-central-1.oss-eu-central-1-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh ; ./logtail.sh install eu-central-1
中东东部 1（迪拜）	wget http://logtail-release-me-east-1.oss-me-east-1-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install me-east-1
英国（伦敦）	wget http://logtail-release-eu-west-1.oss-eu-west-1-internal.aliyuncs.com/linux64/logtail.sh

日志服务Project所在的区域	安装命令
	-O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install eu-west-1

公网

数据通过公网写入日志服务，占用公网带宽。适用于自建IDC和其他云厂商服务器。



说明：

日志服务无法获取ECS之外服务器的属主信息，请在安装Logtail后手动配置用户标识（AliUid，参见 为非本账号ECS、自建IDC配置AliUid），否则 Logtail心跳异常、无法收集日志。

1. 根据日志服务Project所在区域选择安装参数。

   安装命令中的${your_region_name}表示日志服务Project所在区域，根据安装参数选择正确参数，如华东一区域的安装参数为cn-hangzhou。

2. 替换参数后执行安装命令。

   替换参数${your_region_name}后，执行安装命令。

```
wget http://logtail-release-${your_region_name}.oss-${your_region_name}.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install ${your_region_name}-internet
```

   您也可以根据日志服务Project所在的区域直接执行下述对应的命令进行安装：

日志服务Project所在的区域	安装命令
华东 1（杭州）	wget http://logtail-release-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-hangzhou-internet
华东 2（上海）	wget http://logtail-release-cn-shanghai.oss-cn-shanghai.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-shanghai-internet
华北 1（青岛）	wget http://logtail-release-cn-qingdao.oss-cn-qingdao.aliyuncs.com/linux64/logtail.sh -O

日志服务Project所在的区域	安装命令
	logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-qingdao-internet
华北 2（北京）	wget http://logtail-release-cn-beijing.oss-cn-beijing.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-beijing-internet
华北 3（张家口）	wget http://logtail-release-cn-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-zhangjiakou-internet
华北 5（呼和浩特）	wget http://logtail-release-cn-huhehaote.oss-cn-huhehaote.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-huhehaote-internet
华南 1（深圳）	wget http://logtail-release-cn-shenzhen.oss-cn-shenzhen.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-shenzhen-internet
西南 1（成都）	wget http://logtail-release-cn-chengdu.oss-cn-chengdu.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-chengdu-internet
香港	wget http://logtail-release-cn-hongkong.oss-cn-hongkong.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-hongkong-internet
美国西部 1（硅谷）	wget http://logtail-release-us-west-1.oss-us-west-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install us-west-1-internet
美国东部 1（弗吉尼亚）	wget http://logtail-release-us-east-1.oss-us-east-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install us-east-1-internet
亚太东南 1（新加坡）	wget http://logtail-release.oss-cn-hangzhou.aliyuncs.com/linux64/logtail.sh -O logtail.sh;

日志服务Project所在的区域	安装命令
	chmod 755 logtail.sh; sh logtail.sh install ap-southeast-1-internet
亚太东南 2（悉尼）	wget http://logtail-release-ap-southeast-2.oss-ap-southeast-2.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install ap-southeast-2-internet
亚太东南 3（吉隆坡）	wget http://logtail-release-ap-southeast-3.oss-ap-southeast-3.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install ap-southeast-3-internet
亚太东南 5（雅加达）	wget http://logtail-release-ap-southeast-5.oss-ap-southeast-5.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install ap-southeast-5-internet
亚太东北 1（日本）	wget http://logtail-release-ap-northeast-1.oss-ap-northeast-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install ap-northeast-1-internet
欧洲中部 1（法兰克福）	wget http://logtail-release-eu-central-1.oss-eu-central-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install eu-central-1-internet
中东东部 1（迪拜）	wget http://logtail-release-me-east-1.oss-me-east-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install me-east-1-internet
亚太南部 1（孟买）	wget http://logtail-release-ap-south-1.oss-ap-south-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install ap-south-1-internet
英国（伦敦）	wget http://logtail-release-eu-west-1.oss-eu-west-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh

日志服务Project所在的区域	安装命令
	; chmod 755 logtail.sh; ./logtail.sh install eu-west-1-internet

全球加速

如果您的服务器分布在海外各地的自建机房、或者来自海外云厂商，使用公网传输数据可能会出现网络延迟高、传输不稳定等问题，可以通过全球加速传输数据。全球加速利用阿里云CDN边缘节点进行日志采集加速，相对公网采集在网络延迟、稳定性上具有很大优势。

1. 根据日志服务Project所在区域选择安装参数。

   安装命令中的*${your_region_name}*表示日志服务Project所在区域，根据安装参数选择正确参数，如华东一区域的安装参数为cn-hangzhou。

2. 替换参数后执行安装命令。

   替换参数${your_region_name}后，执行安装命令。

```
wget http://logtail-
release-${your_region_name}.oss-${your_region_name}.aliyuncs.com/
linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh
 install ${your_region_name}-acceleration
```

   您也可以根据日志服务Project所在的区域直接执行下述对应的命令进行安装：

华北 2（北京）	wget http://logtail-release-cn-beijing.oss-cn-beijing.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-beijing-acceleration
华北 1（青岛）	wget http://logtail-release-cn-qingdao.oss-cn-qingdao.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-qingdao-acceleration
华东 1（杭州）	wget http://logtail-release-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-hangzhou-acceleration
华东 2（上海）	wget http://logtail-release-cn-shanghai.oss-cn-shanghai.aliyuncs.com/linux64/logtail.sh -O

	logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-shanghai-acceleration
华南 1（深圳）	wget http://logtail-release-cn-shenzhen.oss-cn-shenzhen.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-shenzhen-acceleration
华北 3（张家口）	wget http://logtail-release-cn-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-zhangjiakou-acceleration
华北 5（呼和浩特）	wget http://logtail-release-cn-huhehaote.oss-cn-huhehaote.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-huhehaote-acceleration
西南 1（成都）	wget http://logtail-release-cn-chengdu.oss-cn-chengdu.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-chengdu-acceleration
香港	wget http://logtail-release-cn-hongkong.oss-cn-hongkong.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-hongkong-acceleration
美国西部 1（硅谷）	wget http://logtail-release-us-west-1.oss-us-west-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install us-west-1-acceleration
美国东部 1（弗吉尼亚）	wget http://logtail-release-us-east-1.oss-us-east-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install us-east-1-acceleration
亚太东南 1（新加坡）	wget http://logtail-release-ap-southeast-1.oss-ap-southeast-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install ap-southeast-1-acceleration
亚太东南 2（悉尼）	wget http://logtail-release-ap-southeast-2.oss-ap-southeast-2.aliyuncs.com/linux64/logtail.sh -O

	logtail.sh; chmod 755 logtail.sh; ./logtail.sh install ap-southeast-2-acceleration
亚太东南 3（吉隆坡）	wget http://logtail-release-ap-southeast-3.oss-ap-southeast-3.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install ap-southeast-3-acceleration
亚太东南 5（雅加达）	wget http://logtail-release-ap-southeast-5.oss-ap-southeast-5.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install ap-southeast-5-acceleration
亚太东北 1（日本）	wget http://logtail-release-ap-northeast-1.oss-ap-northeast-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install ap-northeast-1-acceleration
欧洲中部 1（法兰克福）	wget http://logtail-release-eu-central-1.oss-eu-central-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install eu-central-1-acceleration
中东东部 1（迪拜）	wget http://logtail-release-me-east-1.oss-me-east-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install me-east-1-acceleration
亚太南部 1（孟买）	wget http://logtail-release-ap-south-1.oss-ap-south-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install ap-south-1-acceleration
英国（伦敦）	wget http://logtail-release-eu-west-1.oss-eu-west-1.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install eu-west-1-acceleration

ECS 金融云

请根据金融云区域执行对应的安装命令：

金融云 华东 1（杭州）	wget http://logtail-release-hz-finance.oss-cn-hzjbp-b-internal.aliyuncs.com/linux64/logtail.sh -O

	`logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-hangzhou-finance`
金融云 华东 2（上海）	`wget http://logtail-release-sh-finance.oss-cn-shanghai-finance-1-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-shanghai-finance`
金融云 华南 1（深圳）	`wget http://logtail-release-sz-finance.oss-cn-shenzhen-finance-1-internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install cn-shenzhen-finance`

查看Logtail版本

Logtail会将版本信息记录在`/usr/local/ilogtail/app_info.json`中的`logtail_version`字段，例如：

```
$cat /usr/local/ilogtail/app_info.json
{
   "UUID" : "0DF18E97-0F2D-486F-B77F-*********",
   "hostname" : "david*******",
   "instance_id" : "F4FAFADA-F1D7-11E7-846C-00163E30349E_*********
_1515129548",
   "ip" : "**********",
   "logtail_version" : "0.16.0",
   "os" : "Linux; 2.6.32-220.23.2.ali1113.el5.x86_64; #1 SMP Thu Jul 4
 20:09:15 CST 2013; x86_64",
   "update_time" : "2018-01-05 13:19:08"
}
```

升级Logtail

升级Logtail和安装的操作步骤一致，如果您已经安装Logtail，再次执行安装命令时将自动卸载并安装最新版本的Logtail。

手动启动和停止Logtail

- 启动

  以管理员身份执行：

  ```
  /etc/init.d/ilogtaild start
  ```

- 停止

以管理员身份执行：

```
/etc/init.d/ilogtaild stop
```

卸载Logtail

执行以下命令，下载安装器**logtail.sh**并卸载Logtail。

```
wget http://logtail-release-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com/
linux64/logtail.sh -O logtail.sh
chmod 755 logtail.sh; ./logtail.sh uninstall
```

4.3.2 Windows

Logtail客户端是日志服务提供的日志采集Agent，请参考本文档，在Windows服务器上安装Logtail客户端。

支持的系统

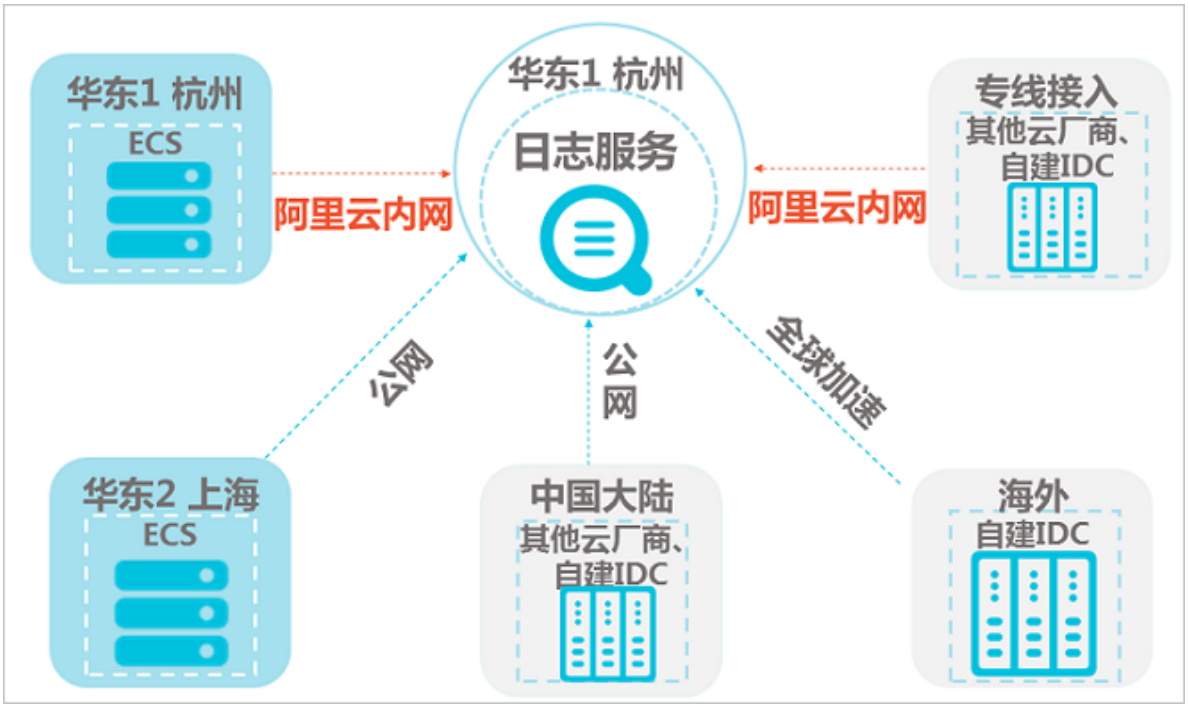
支持Windows Server2003（含）以后 32/64 位系统，例如：

- Windows 7 (Client) 32bit
- Windows 7 (Client) 64bit
- Windows Server 2003 32bit
- Windows Server 2003 64bit
- Windows Server 2008 32bit
- Windows Server 2008 64bit
- Windows Server 2012 64bit

前提条件

1. 已拥有一台及以上的服务器。
2. 已根据服务器类型和所在Region，确定日志采集流量的网络类型。详细说明请参考选择网络。

图 **4-5:** 选择网络



安装Logtail

1. 下载安装包。

   单击下载*Logtail*安装包。

2. 解压缩 `logtail_installer.zip` 到当前目录。

3. 根据服务器类型和所在区域选择网络类型后，按照日志服务所在区域安装Logtail。

   打开Windows Powershell或cmd进入 `logtail_installer` 目录，并根据地域和网络环境执行相应的安装命令。

   安装命令：

区域	阿里云内网（经典网络、VPC）	公网	全球加速
华北 1（青岛）	.\logtail_installer.exe install cn-qingdao	.\logtail_installer.exe install cn-qingdao-internet	.\logtail_installer.exe install cn-qingdao-acceleration
华北 2（北京）	.\logtail_installer.exe install cn-beijing	.\logtail_installer.exe install cn-beijing-internet	.\logtail_installer.exe install cn-beijing-acceleration

区域	阿里云内网（经典网络、VPC）	公网	全球加速
华北 3（张家口）	.\logtail_installer.exe install cn-zhangjiakou	.\logtail_installer.exe install cn-zhangjiakou-internet	.\logtail_installer.exe install cn-zhangjiakou-acceleration
华北 5（呼和浩特）	.\logtail_installer.exe install cn-huhehaote	.\logtail_installer.exe install cn-huhehaotei nternet	.\logtail_installer.exe install cn-huhehaote-acceleration
华东 1（杭州）	.\logtail_installer.exe install cn-hangzhou	.\logtail_installer.exe install cn-hangzhou-internet	.\logtail_installer.exe install cn-hangzhou-acceleration
华东 2（上海）	.\logtail_installer.exe install cn-shanghai	.\logtail_installer.exe install cn-shanghai-internet	.\logtail_installer.exe install cn-shanghai-acceleration
华南 1（深圳）	.\logtail_installer.exe install cn-shenzhen	.\logtail_installer.exe install cn-shenzhen-internet	.\logtail_installer.exe install cn-shenzhen-acceleration
西南 1（成都）	.\logtail_installer.exe install cn-chengdu	.\logtail_installer.exe install cn-chengdu-internet	.\logtail_installer.exe install cn-chengdu-acceleration
香港	.\logtail_installer.exe install cn-hongkong	.\logtail_installer.exe install cn-hongkong-internet	.\logtail_installer.exe install cn-hongkong-acceleration
美国西部 1（硅谷）	.\logtail_installer.exe install us-west-1	.\logtail_installer.exe install us-west-1-internet	.\logtail_installer.exe install us-west-1-acceleration
美国东部 1（弗吉尼亚）	.\logtail_installer.exe install us-east-1	.\logtail_installer.exe install us-east-1-internet	.\logtail_installer.exe install us-east-1-acceleration
亚太东南 1（新加坡）	.\logtail_installer.exe install ap-southeast-1	.\logtail_installer.exe install ap-southeast-1-internet	.\logtail_installer.exe install ap-southeast-1-acceleration
亚太东南 2（悉尼）	.\logtail_installer.exe install ap-southeast-2	.\logtail_installer.exe install ap-southeast-2-internet	.\logtail_installer.exe install ap-southeast-2-acceleration

区域	阿里云内网（经典网络、VPC）	公网	全球加速
亚太东南 3（吉隆坡）	.\logtail_installer.exe install ap-southeast-3	.\logtail_installer.exe install ap-southeast-3-internet	.\logtail_installer.exe install ap-southeast-3-acceleration
亚太东南 5（雅加达）	.\logtail_installer.exe install ap-southeast-5	.\logtail_installer.exe install ap-southeast-5-internet	.\logtail_installer.exe install ap-southeast-5-acceleration
亚太南部 1（孟买）	.\logtail_installer.exe install ap-south-1	.\logtail_installer.exe install ap-south-1-internet	.\logtail_installer.exe install ap-south-1-acceleration
亚太东北 1（日本）	.\logtail_installer.exe install ap-northeast-1	.\logtail_installer.exe install ap-northeast-1-internet	.\logtail_installer.exe install ap-northeast-1-acceleration
欧洲中部 1（法兰克福）	.\logtail_installer.exe install eu-central-1	.\logtail_installer.exe install eu-central-1-internet	.\logtail_installer.exe install eu-central-1-acceleration
中东东部 1（迪拜）	.\logtail_installer.exe install me-east-1	.\logtail_installer.exe install me-east-1-internet	.\logtail_installer.exe install me-east-1-acceleration
英国（伦敦）	.\logtail_installer.exe install eu-west-1	.\logtail_installer.exe install eu-west-1-internet	.\logtail_installer.exe install eu-west-1-acceleration
金融云 华东 1（杭州）	.\logtail_installer.exe install cn-hangzhou-finance	无此类型网络	无此类型网络
金融云 华东 2（上海）	.\logtail_installer.exe install cn-shanghai-finance	无此类型网络	无此类型网络
金融云 华南 1（深圳）	.\logtail_installer.exe install cn-shenzhen-finance	无此类型网络	无此类型网络



说明：

在自建IDC或其他云厂商服务器使用Logtail时，由于日志服务无法获取非本账号下ECS、其他服务器的属主信息，请在安装Logtail后手动配置用户标识（AliUid），否则Logtail心跳异常、无法收集日志。详细说明请参见 为非本账号ECS、自建IDC配置AliUid。

升级Logtail

Windows版Logtail支持自动升级。

手动启动和停止Logtail

打开控制面板中的管理工具，打开服务，找到*LogtailWorker*。

- 手动启动：右键单击停止。
- 停止：右键单击启动。
- 重启：右键单击重新启动。

卸载Logtail

打开 Windows Powershell或cmd进入 `logtail_installer` 目录，执行命令：

```
.\logtail_installer.exe uninstall
```

4.3.3 配置启动参数

本文描述Logtail启动配置参数，如有特殊需求，可以参考本文进行设置。

应用场景

配置Logtail启动配置参数适用于以下场景：

- 需要采集的日志文件数目大，占用大量内存。内存中需要维护每个文件的签名、采集位置、文件名等 meta 信息。
- 日志数据流量大导致CPU占用率高。
- 日志数据量大导致发送到日志服务的网络流量大。

启动配置

- 文件路径

  ```
  /usr/local/ilogtail/ilogtail_config.json
  ```

- 文件格式

  JSON

- 文件示例（只展示部分配置项）

  ```
  {
      ...
      "cpu_usage_limit" : 0.4,
      "mem_usage_limit" : 100,
      "max_bytes_per_sec" : 2097152,
  ```

```
    "process_thread_count" : 1,
    "send_request_concurrency" : 4,
    "buffer_file_num" : 25,
    "buffer_file_size" : 20971520,
    "buffer_file_path" : "",
    ...
}
```

常用配置参数

参数名	参数说明	取值
cpu_usage_limit	CPU 使用阈值，以单核计算。 大部分场景下，极简模式单核处理能力约24MB/s，完整正则模式单核处理能力约12MB/s 。参考信息	double 类型。最小值为0.1，最大值为当前机器的CPU核心数，默认值为2。 若设置为0.4，表示限制 Logtail 的CPU使用为CPU单核的40 %，超出后Logtail自动重启。
mem_usage_limit	常驻内存使用阈值。 如需要采集的distinct文件数目超过1000，请酌情修改上调该阈值。	int 类型，为单位为MB。最小值为128，最大值为当前机器有效内存值，默认值为2048。 若设置为100，表示Logtail的内存使用为100兆字节，超出后Logtail自动重启。
max_bytes_per_sec	Logtail 发送原始数据的流量限制，超过20 MB/s不限流。	int类型，单位为Byte/Sec。取值范围为1024~52428800，默认值为20971520。 若设置为2097152，表示Logtail发送数据的速率为 2MB/s。
process_thread_count	Logtail 处理日志文件写入数据的线程数。一般可以处理极简模式 24MB/s 或完整正则模式 12MB/s 的写入。默认情况下无需调整该参数取值，可以必要的时候适当上调。	int类型，单位为个。取值范围为1~64，默认值为1。
send_request_concurrency	异步并发的个数。Logtail 默认异步发送数据包，如果写入 TPS 很高，可以配置更高的异步并发。 可以按照一个并发支持 0.5MB/s~1MB/s 网络吞吐来计算，具体依据网络延时而定。 说明：	int类型，单位为个。取值范围为1~1000，默认值为20。

参数名	参数说明	取值
	参数过高容易导致网络端口占用过多，此时需调整TCP相关参数。	
buffer_file_num	网络异常，写入配额超限后，Logtail将实时解析后的日志写入本地文件（安装目录下）缓存起来，等待恢复后尝试重新发送服务端。该参数限制缓存文件的最大数目。	int类型，单位为个。取值范围为1~100，默认为 25。
buffer_file_size	该参数用于设置单个缓存文件允许的最大字节数，**buffer_file_num * buffer_file_size**是缓存文件可以实际使用的最大磁盘空间。	int类型，单位为Byte。取值范围为1048576~104857600，默认值为20971520 Byte，即20MB。
buffer_file_path	该参数用于设置缓存文件存放目录，请在修改该参数后，手动将旧缓存目录下名称如 *logtail_buffer_file_** 的文件移动到新缓存目录，以保证 logtail 可以读取到该缓存文件并在发送后进行删除。	默认情况下该参数为空，缓存文件存放于程序安装目录*/usr/local/ilogtail*。
bind_interface	本机绑定的网卡名，例如eth1。只支持Linux版本。	默认情况下该参数为空，自动绑定可用网卡，若配置该参数，Logtail将强制使用该网卡进行日志上传。
check_point_filename	checkpoint文件保存的全路径，用于自定义Logtail的checkpoint保存位置。 建议Docker用户修改此文件保存地址，并将checkpoint所在目录挂载到宿主机，否则容器释放时会因丢失checkpoint信息而产生重复采集。例如Docker中配置**check_point_filename**为 `/data/logtail/check_point.dat`，Docker启动命令增加`-v /data/docker1/logtail:/data/logtail`，将宿主机*/data/docker1/logtail*目录挂载到Docker中的*/data/logtail*目录。	默认情况下为*/tmp/logtail_check_point*。
`user_config_file_path`	采集配置文件保存的全路径，用于自定义Logtail的采集配置保存位置。	默认情况下为进程binary所在目录，文件名为*user_log_config.json*。

参数名	参数说明	取值
	建议Docker用户修改此文件保存地址，并将采集配置所在目录挂载到宿主机，否则容器释放时会因丢失checkpoint信息而产生重复采集。 例如Docker中配置`user_config_file_path`为 *`/data/logtail/user_log_config.json`*，Docker启动命令增加`-v /data/docker1/logtail:/data/logtail`，将宿主机*`/data/docker1/logtail`*目录挂载到Docker中的*`/data/logtail`*目录。	
`discard_old_data`	是否丢弃历史日志，若该值为true，则丢弃距当前时间超过12小时的日志。	bool类型，默认值为true。
`working_ip`	Logtail上报的本机IP地址，若该值为空，则自动从本机获取IP。	IP地址，默认为空。
`working_hostname`	Logtail上报的本机hostname，若为空，则自动获取本机hostname。	字符串类型，默认为空。
`max_read_buffer_size`	每条日志读取最大值，若您的单条日志超过512KB，可调整此选项。	long类型，默认值524288Byte（512KB）。



说明：

- 这里只列出您需要关注的常用启动参数，如 *`ilogtail_config.json`* 内有表格中未列出的参数，会使用默认配置，属于正常情况。
- 请根据需要新增或修改指定配置参数所对应的值，不必要的配置项不需要增加到 *`ilogtail_config.json`*。

修改配置

1. 按需配置 *`ilogtail_config.json`*

   请确认修改配置后，配置内容为合法 JSON。

2. 重启Logtail使配置生效。

```
/etc/init.d/ilogtaild stop
/etc/init.d/ilogtaild start
```

```
/etc/init.d/ilogtaild status
```

4.4 机器组

4.4.1 简介

日志服务通过机器组的方式管理所有需要通过 Logtail 客户端收集日志的服务器。

机器组是包含多台服务器的虚拟分组。如果您有一台以上的服务器，并希望这些服务器使用同样的Logtail配置采集日志，可以将这些服务器加入同一个机器组，并将Logtail配置应用到该机器组。

您可以通过如下两种方法定义一个机器组：

- *IP*地址：在机器组中添加服务器的IP地址，通过IP地址识别服务器。
- 自定义标识：定义属于机器组的一个标识，在对应服务器上配置对应标识进行关联。



说明：

- 其他云厂商服务器、自建IDC、其他账号下的ECS添加到机器组之前，请先在服务器上配置AliUid（用户标识），详细步骤请参考为非本账号*ECS*、自建*IDC*配置*AliUid*。
- 请勿将Windows服务器和Linux服务器添加到同一机器组中。

IP地址机器组

您可以通过添加服务器IP地址的方式，直接将多台服务器添加到一个机器组中，为其统一进行Logtail配置。

- 若您使用ECS服务器，且没有绑定过hostname、没有更换过网络类型，可以在机器组中配置ECS服务器的私网IP地址。
- 其他情况下，请在机器组中配置Logtail自动获取到的IP地址，该IP地址记录在服务器文件*app_info.json*的ip字段中。



说明：

*app_info.json*是记录Logtail内部信息的文件，其中包括Logtail自动获取到的IP地址，手动修改该文件的ip字段不能改变Logtail获取的IP地址。

Logtail自动获取服务器IP地址的方式：

- 如果已在服务器文件*/etc/hosts*中设置了主机名与IP地址绑定，则自动获取绑定的IP地址。
- 如果没有设置主机名绑定，会自动获取本机的第一块网卡的IP地址。



说明：

数据采集是否使用阿里云内网，与机器组中填写的IP地址是否为私网IP地址无关。如果您的服务器是阿里云ECS云服务器，并且安装Logtail时选择了阿里云内网（经典网络/VPC）模式，只有这种情况下您的日志数据才可以通过阿里云内网采集到日志服务。

创建IP地址机器组，请参考创建IP地址机器组。

自定义标识机器组

除IP地址外，您还可以使用自定义标识（userdefined-id）来动态定义机器组。

自定义标识机器组在以下场景中具有明显优势：

- VPC等自定义网络环境中，可能出现不同服务器IP地址冲突的问题，导致服务端无法管理Logtail。使用自定义标识可以避免此类情况的发生。
- 多台服务器通过同一个自定义标识实现机器组弹性伸缩。您只需为新增的服务器配置相同的自定义标识，服务端可自动识别，并将其添加至机器组中。

通常情况下，系统由多个模块组成，每个模块都可以进行单独的水平扩展，即支持添加多台服务器。为每个模块分别创建机器组，可以达到分类采集日志的目的。因此需要为每个模块分别创建自定义标识，并在各个模块的服务器上配置各自所属的机器组标识。例如常见网站分为前端 HTTP 请求处理模块、缓存模块、逻辑处理模块和存储模块，其自定义标识可以分别定义为http_module、cache_module、logic_module和store_module。

创建自定义标识机器组，请参考创建用户自定义标识机器组。

4.4.2 创建IP地址机器组

日志服务支持创建IP地址机器组。将Logtail获取到的服务器IP地址添加进IP地址机器组中，则可以使用同样的Logtail配置采集这些服务器的日志。

前提条件

- 已创建Project和Logstore。
- 已有一台及以上的服务器，如果是阿里云ECS云服务器，请确保ECS和当前日志服务 Project 在同一阿里云地域下。
- 已为服务器安装了Logtail。详细步骤请参考*Linux*和*Windows*。
- 其他云厂商服务器、自建IDC、其他阿里云账号下的ECS请先配置AliUid（用户标识），详细步骤请参考为非本账号*ECS*、自建*IDC*配置*AliUid*。

背景信息

数据采集是否使用阿里云内网，与机器组中填写的**IP**地址是否为私网**IP**地址无关。如果您的服务器是阿里云ECS云服务器，ECS和当前日志服务Project在同一阿里云地域，并且安装Logtail时选择了阿里云内网（经典网络/VPC）模式，只有这种情况下您的日志数据才可以通过阿里云内网采集到日志服务。

操作步骤

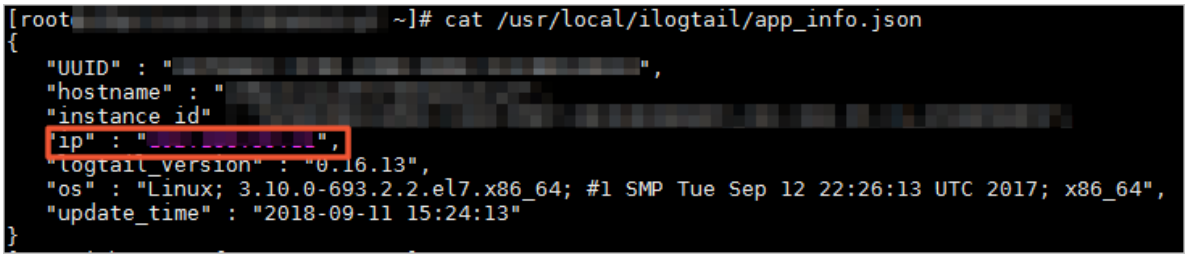
1. 查看服务器IP地址，即Logtail自动获取到的IP地址。

   Logtail自动获取到的**IP**地址记录在文件***app_info.json***的**ip**字段中。

   在安装了Logtail的服务器上查看*app_info.json*文件，路径为：

   - Linux：*/usr/local/ilogtail/app_info.json*
   - Windows x64：*C:\Program Files (x86)\Alibaba\Logtail\app_info.json*
   - Windows x32：*C:\Program Files\Alibaba\Logtail\app_info.json*

   例如，在Linux中查看服务器IP地址：

2. 登录日志服务控制台，单击Project名称。
3. 单击左侧导航栏的 **LogHub - 实时采集** > **Logtail机器组** 进入该项目的 **机器组列表** 页面。
4. 单击右上角的**创建机器组**。

   您也可以在数据接入向导中创建采集配置后，于 **应用到机器组** 页面，单击 **创建机器组**。
5. 创建机器组。

   a) 填写机器组名称。

   名称只能包含小写字母、数字、连字符（-）和下划线（_）且必须以小写字母或数字开头和结尾，长度为3~128字节。

   b) 填写机器组名称。

   > **说明：**

不支持修改机器组名称，请谨慎填写。

c) 选择机器组标识为**IP 地址**。

d) 填写**IP地址**。

请填写*1*中获取到的服务器IP地址。

说明：

- 请确保您已按照*1*获取了服务器IP地址。
- 机器组中存在多台服务器时，IP地址之间请用换行符分割。
- 请勿将Windows服务器和Linux服务器添加到同一机器组中。



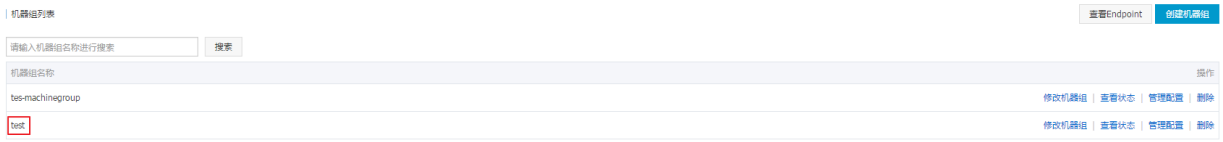
6. （可选）填写 机器组**Topic**。

机器组Topic的详细信息请参考生成主题。

7. 单击 确定。

预期结果

您可以在机器组列表中查看刚创建的机器组。



4.4.3 创建用户自定义标识机器组

除IP地址外，您还可以使用用户自定义标识（Custom ID）来动态定义机器组。

自定义标识机器组在以下场景中具有明显优势：

- VPC等自定义网络环境中，可能出现不同机器IP地址冲突的问题，导致服务端无法管理Logtail。使用自定义标识可以避免此类情况的发生。
- 多台通过同一服务器个自定义标识实现机器组弹性伸缩。您只需为新增的服务器同的自定义标识，服务端可自动识别，并将其添加至机器组中。

操作步骤

1. 在服务器上设置用户自定义标识。
   - **Linux Logtail**

     通过文件`/etc/ilogtail/user_defined_id`来设置用户自定义标识。

     例如，设置用户自定义标识如下：

     ```
     # vim /etc/ilogtail/user_defined_id
     ```

     在该文件中输入`userdefined`。
   - **Windows Logtail**

     通过文件`C:\LogtailData\user_defined_id`来设置用户自定义标识。

     例如，设置用户自定义标识如下：

     ```
     C:\LogtailData>more user_defined_id
     userdefined_windows
     ```

   说明：

   - 同一机器组中不允许同时存在Linux和Windows服务器，请勿在Linux和Windows服务器上配置同样的用户自定义标识。

- 一个服务器可配置多个用户自定义标识，标识之间以换行符分割。
- 若目录 */etc/ilogtail/*、*C:\LogtailData*或文件*/etc/ilogtail/user_defined_id*、*C:\LogtailData\user_defined_id*不存在，请手动创建。

2. 创建机器组。

   a. 登录日志服务控制台，单击Project名称。

   b. 单击左侧导航栏中的**Logtail机器组**。

   c. 在**机器组列表**页面单击右上角的**创建机器组**。

   d. 填写机器组配置。

   - **机器组名称**：填写机器组名称。

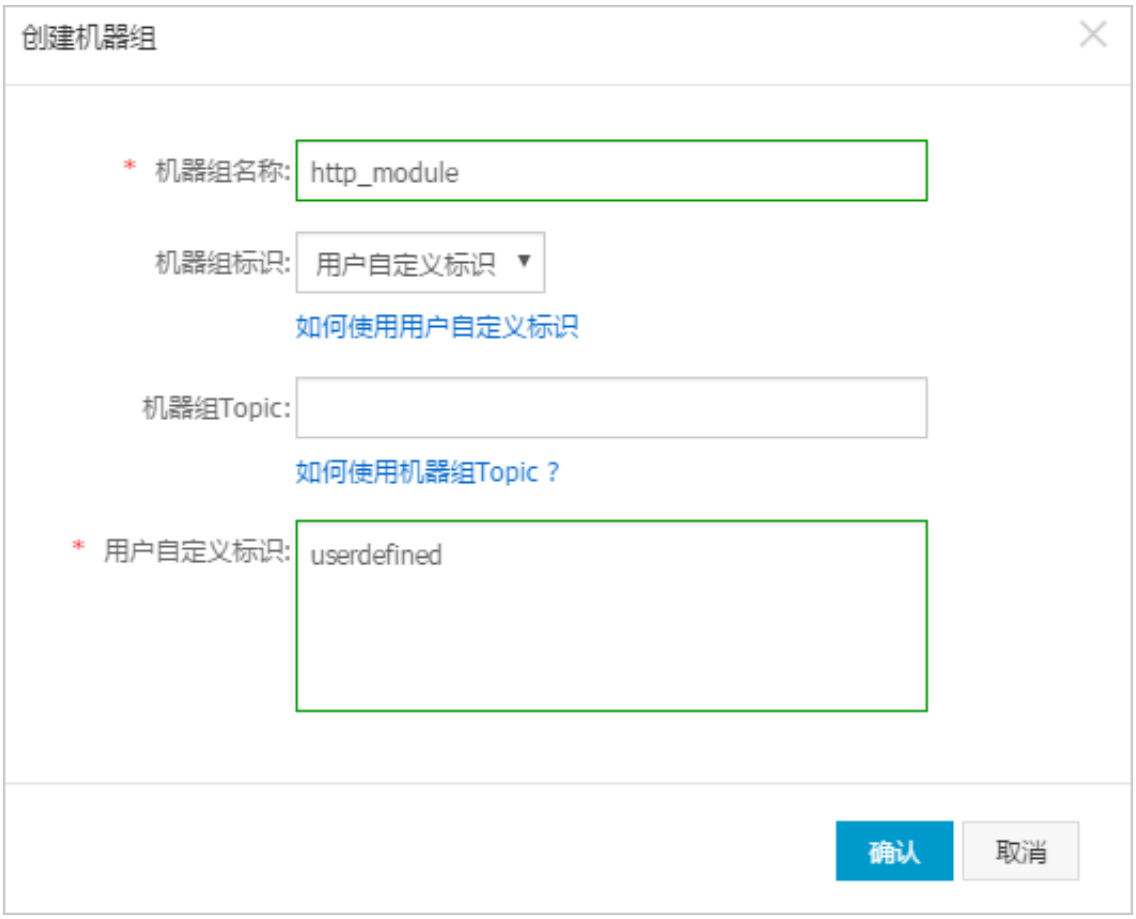
     机器组名称只能包含小写字母、数字、连字符（-）和下划线（_）且必须以小写字母或数字开头和结尾，长度为3~128字节。

     说明：

     不支持修改机器组名称，请谨慎填写。

   - **机器组标识**：选择用户自定义标识。
   - （可选）**机器组Topic**：填写机器组Topic，详细信息请参考生成主题。
   - **用户自定义标识**：填写步骤一中配置的用户自定义标识。



e. 单击**确认**结束配置。

> **说明：**
> 需要扩容服务器时，只需要在新服务器上设置用户自定义标识即可。

3. 查看机器组状态。

   在**机器组列表**页面，单击目标机器组**操作**列的**查看状态**，可以查看使用相同用户自定义标识的服务器列表及其心跳状态。



禁用用户自定义标识

如果想恢复使用服务器IP作为标识，请删除user_defined_id文件，1分钟之内即可生效。

- **Linux系统**

```
rm -f /etc/ilogtail/user_defined_id
```

- **Windows系统**

```
del C:\LogtailData\user_defined_id
```

生效时间

新增、删除、修改user_defined_id文件后，默认情况下，1分钟之内即可生效。

如需立即生效，请执行以下命令重启Logtail：

- **Linux系统**

```
/etc/init.d/ilogtaild stop
/etc/init.d/ilogtaild start
```

- **Windows系统**

  Windows控制面板 > 管理工具 > 服务，在服务列表中右键单击**LogtailWorker**服务，选择重新启动以使配置生效。

示例

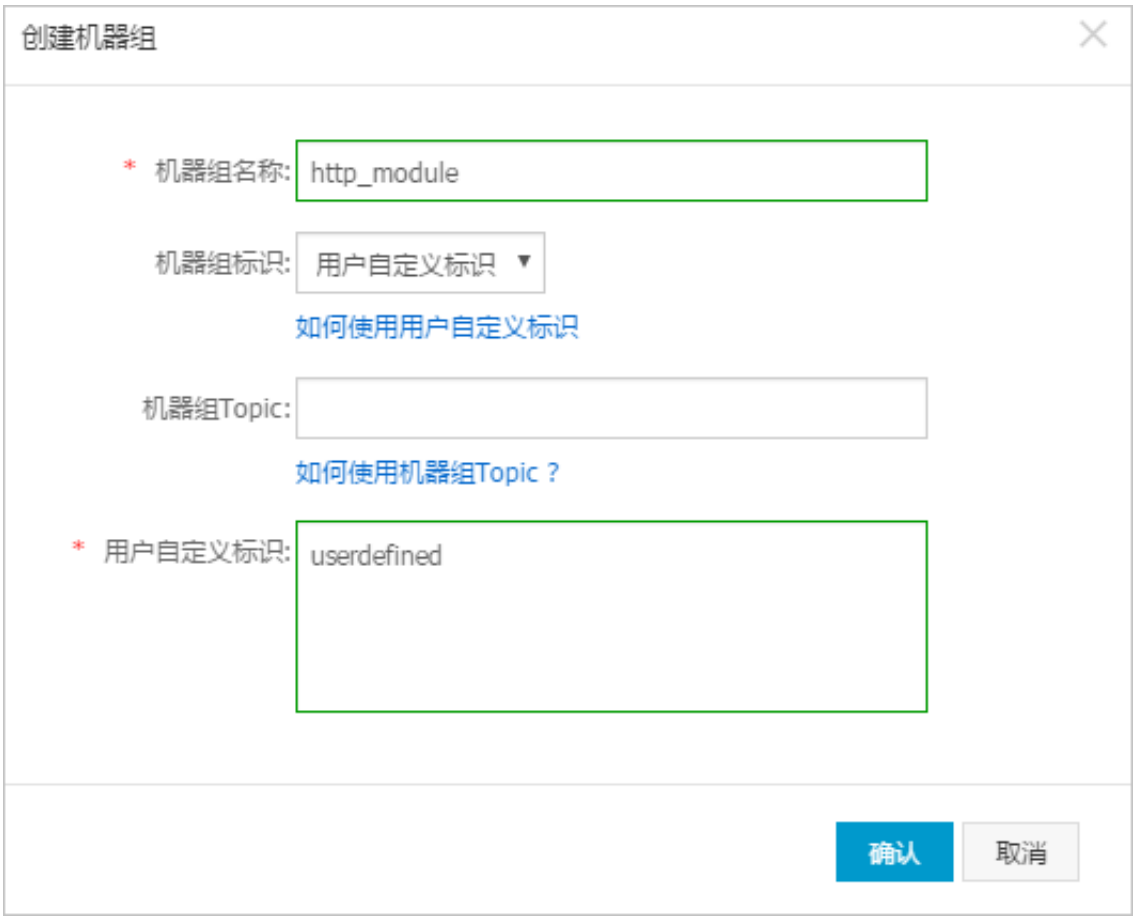
系统通常由多个模块组成，每个模块可以包含多台服务器。比如常见网站分为前端HTTP请求处理模块、缓存模块、逻辑处理模块和存储模块，每个模块可以进行单独的水平扩展，因此在新增服务器时需要能够对其进行实时日志采集。

1. 创建自定义标识。

   安装完成Logtail客户端后，为服务器开启用户自定义标识。对于示例场景中的模块可以分别分成4类机器标识：http_module、cache_module、logic_module和store_module。

2. 创建机器组。

   创建机器组时，**用户自定义标识**请填写机器组名称对应的用户自定义标识。http_module机器组如下图所示：



3. 可以在机器组**查看状态**中查看使用相同自定义标识的服务器列表及其心跳状态。



4. 若前端模块增加服务器10.1.1.3，只需在新服务器上开启用户自定义标识。成功执行操作后可以在机器组查看状态中看到新增服务器。



4.4.4 为非本账号ECS、自建IDC配置AliUid

其他账号购买的ECS、其他云厂商的服务器和自建IDC在安装Logtail之后，需要配置AliUid作为用户标识，才能加入机器组开始采集日志。

背景信息

如果您需要通过Logtail采集日志的服务器并非阿里云ECS，或非本账号购买的ECS时，需要在服务器上安装Logtail之后，配置AliUid作为用户标识，证明这台服务器有权限被该账号访问、采集日志。否则在机器组中会显示服务器心跳失败，无法采集数据到日志服务。

前提条件

- 需要采集日志的服务器为其他账号购买的ECS、其他云厂商的服务器和自建IDC。
- 已在服务器上安装Logtail客户端。

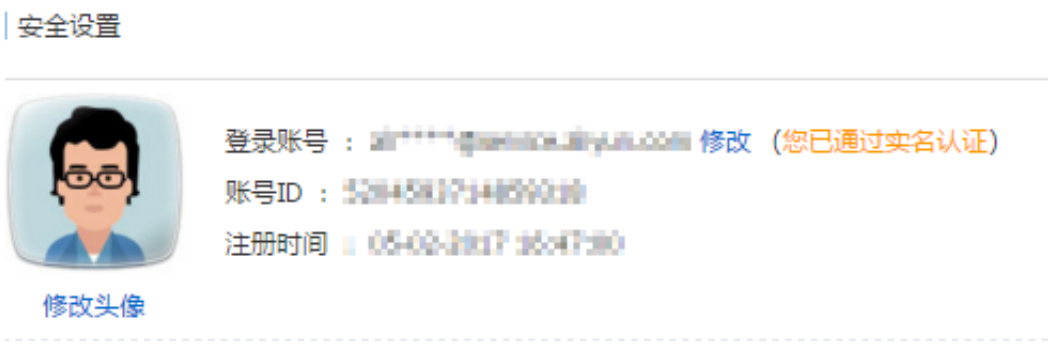
  详细步骤请参见*Linux* 和*Windows*。

操作步骤

1. 查看阿里云账号ID，即AliUid。

   登陆账号管理页面，查看日志服务Project所属账号的AliUid。

   图 **4-6: 查看AliUid**

2. 登录服务器，配置AliUid作为用户标识。

   - **Linux系统**

     创建AliUid同名文件到 */etc/ilogtail/users* 目录，如目录不存在请手动创建目录。一台服务器上可以配置多个AliUid，例如：

     ```
     touch /etc/ilogtail/users/1559122535028493
     touch /etc/ilogtail/users/1329232535020452
     ```

     当不需要 Logtail 收集数据到该用户的日志服务 Project 后，可删除AliUid用户标识：

     ```
     rm /etc/ilogtail/users/1559122535028493
     ```

   - **Windows系统**

创建AliUid同名文件到目录 `C:\LogtailData\users`以配置AliUid作为用户标识。如需删除AliUid用户标识，请直接删除此文件。

例如，`C:\LogtailData\users\1559122535028493`。



说明：

- 服务器置AliUid作为用户标识后，表示该账号有权限通过 Logtail 收集该服务器日志数据。请及时清理服务器上多余的账号AliUid用户标识文件。
- 新增、删除AliUid用户标识后，1分钟之内即可生效。

后续操作

创建*IP*地址机器组或创建自定义标识机器组

4.4.5 管理采集配置

Logtail 客户端可以帮助日志服务用户简单得通过控制台就能收集 ECS 云主机上的日志。除了安装 Logtail 客户端外（参见 *Linux* 和*Windows*），为 Logtail 客户端创建日志收集配置也非常关键。您可以通过日志库列表给相应日志库创建、修改 Logtail 配置。

创建 Logtail 配置

有关如何通过日志服务云控制台创建 Logtail 配置，参见使用 采集文本日志 和 *Syslog*。

查看 Logtail 配置列表

1. 登录日志服务管理控制台。
2. 选择所需的项目，单击项目名称，进入**Logstore列表**。
3. 在 **Logstore列表** 页面，单击日志收集模式列下的 Logtail配置 管理，进入 **Logtail配置列表** 页面。

   该页面列出了指定Logtail对应的所有配置，其中包含三部分内容：配置名称、数据来源和配置详情，其中当数据来源为 **文本文件** 时，配置详情展示了文件路径和文件名称，如下图所示。

图 4-7: Logtail配置列表



说明：

一个文件只能被一个配置收集。

修改 Logtail 配置

1. 登录日志服务管理控制台。
2. 选择所需的项目，单击项目名称。
3. 在 **Logstore列表** 页面，单击管理，进入 **Logtail配置列表** 页面。
4. 单击需要修改的 Logtail 配置的名称。

   您可以修改日志的收集模式并重新指定应用到的机器组。整个配置修改的流程和创建完全相同。

删除 Logtail 配置

1. 登录日志服务管理控制台。
2. 选择所需的项目，单击项目名称。
3. 在 **Logstore列表** 页面，单击日志收集模式列下的 Logtail配置 管理，进入 **Logtail配置列表** 页面。
4. 选择需要删除的 Logtail 配置并单击右侧的 删除。

   删除成功后即会将该配置与之前应用机器组解除绑定，Logtail 也会停止收集该配置对应的日志文件内容。

说明：

删除指定 logstore 前必须删除其对应的所有 Logtail 配置。

4.4.6 管理机器组

日志服务通过机器组的方式管理所有需要通过Logtail客户端收集日志的ECS云服务器。您可以通过日志服务项目列表进入该项目的 **机器组列表** 页面。您可以通过日志服务创建机器组，查看机器组列表，修改机器组，查看机器组状态，管理配置，删除机器组，使用机器组标识。

创建机器组

您可以通过如下两种方法定义一个机器组：

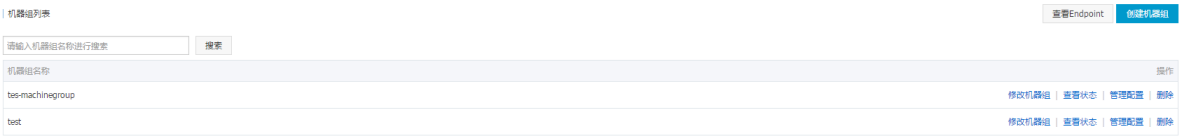
- IP：定义机器组名称并添加一组机器的内网IP。详细步骤请参考创建*IP*地址机器组。
- 标识：定义属于机器组一个标识，在对应机器上配置对应标识进行关联。详细步骤请参考创建用户自定义标识机器组。

查看机器组列表

1. 登录日志服务控制台。
2. 单击Project名称进入**Logstore列表**，单击左侧导航栏的**机器组管理** 进入 **机器组管理** 页面。

   您可以查看该项目下的所有机器组。

   图 **4-8:** 查看机器组列表



修改机器组

在创建完机器组后，您可以随时调整该机器组内的云主机列表。

> **说明：**
> 机器组名称在创建时选定后就不可以更改。

1. 登录日志服务控制台。
2. 单击Project名称进入**Logstore列表**，单击左侧导航栏的**机器组管理** 进入 **机器组管理** 页面。

   您可以查看该项目下的所有机器组。
3. 选择需要修改的机器组，单击 **修改**。
4. 修改机器组的配置信息并单击 **确定**。

> **说明：**
> 不支持修改机器组名称。

图 **4-9:** 修改机器组



查看状态

为验证 Logtail 客户端已经在机器组内的所有云主机安装成功，您可以查看 Logtail 客户端的心跳信息。

> **说明：**
> 机器加入机器组后，机器组的心跳状态响应有2分钟左右延时。请耐心等待2分钟以上再查看心跳状态。

1. 登录日志服务控制台。
2. 单击Project名称进入**Logstore**列表，单击左侧导航栏的机器组管理 进入 机器组管理 页面。

3. 选择机器组，单击 状态。

   - 心跳状态为**OK**表示所有云主机上的 Logtail 客户端都安装成功，Logtail与日志服务连接正常。
   - 心跳状态为**FAIL**表示Logtail连接异常。如果服务器心跳状态始终显示为**FAIL**，请按照页面提示及文档*Logtail* 机器无心跳进行排查。如自查仍无法解决问题，可通过工单寻求帮助。

   图 **4-10:** 查看机器组状态



管理配置

日志服务利用机器组管理所有需要收集日志的云主机，这其中的一个重要管理项目就是 Logtail 客户端的收集配置（请参考采集文本日志和 *Syslog*）。您可以通过给一个机器组应用、删除 Logtail 配置来决定每台云主机上的 Logtail 收集哪些日志，如何解析这些日志，发送日志到哪个日志库等。

1. 登录日志服务控制台。
2. 单击Project名称进入**Logstore列表**，单击左侧导航栏的**机器组管理** 进入 **机器组管理** 页面。
3. 选择机器组，单击 **应用配置**。
4. 您可以选择所需的配置并单击**确定** 来修改应用到机器组的 Logtail 配置。

   当添加 Logtail 配置时，该 Logtail 配置就会下发到机器组内云主机的 Logtail 客户端。当移除 Logtail 配置时，该 Logtail 配置会从 Logtail 客户端移除。

图 4-11: 管理机器组配置



删除机器组

1. 登录日志服务控制台。
2. 单击Project名称进入**Logstore**列表，单击左侧导航栏的机器组管理 进入 机器组管理 页面。
3. 选择机器组，单击 删除。
4. 在确认对话框里，单击 确定。

图 4-12: 删除机器组



4.5 文本日志

4.5.1 采集文本日志

Logtail客户端可以帮助日志服务用户简单地通过控制台收集ECS云服务器或您本地服务器上的文本日志。

前提条件

- 设置使用Logtail收集日志前，您需要安装Logtail。Logtail支持Windows和 Linux两大操作系统，安装方法参见*Linux*和*Windows*。
- 采集ECS或本地服务器日志，请确保您已开启了80端口和443端口。

使用限制

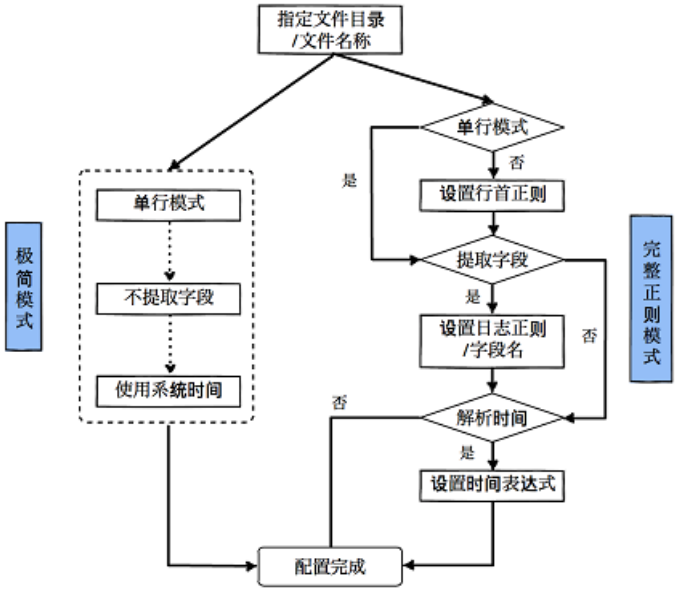
- 一个文件只能被一个配置收集。如果需要采集多份，建议以软链接形式实现。例如`/home/log/nginx/log`下需要采集两份，则其中一个配置原始路径，另外创建一个该文件夹的软链接`ln -s /home/log/nginx/log /home/log/nginx/link_log`，另一个配置软链接路径即可。
- Logtail客户端支持的操作系统可参考*Logtail*简介。
- 经典网络或者VPC下的ECS必须和日志服务Project在相同地域；如果您的日志源数据通过公网传输（类似IDC用法），可以根据地域说明选择日志服务Project的地域。

  经典网络或者VPC下的ECS必须和日志服务Project在相同地域；如果您的日志源数据通过公网传输（类似IDC用法），可以根据实际需求选择日志服务Project的地域。

日志收集配置流程

通过控制台配置Logtail收集文本日志，可以通过极简模式、分隔符模式、JSON 模式、完整正则模式等方式收集日志进行设置。以极简模式和完整正则模式为例，配置流程如下。

图 4-13: 配置流程



操作步骤

1. 在日志服务管理控制台单击目标项目，进入**Logstore**列表。
2. 选择目标Logstore，并单击Logstore名称右侧的**数据接入向导**图标，进入配置数据接入流程。
3. 选择数据类型。

   单击**自定义数据**中的**文本文件**，并单击下一步，进入**数据源设置**界面。
4. 指定**配置名称**。

   配置名称只能包含小写字母、数字、连字符（-）和下划线（_），且必须以小写字母和数字开头和结尾，长度为3~63字节。

   > 说明：
   > 配置名称设置后不可修改。
5. 指定日志的目录和文件名。

   目录结构支持完整路径和通配符两种模式。

   > 说明：
   > 目录通配符只支持 *和? 两种。

   日志文件名支持完整文件名和通配符两种模式，文件名规则请参考*Wildcard matching*。

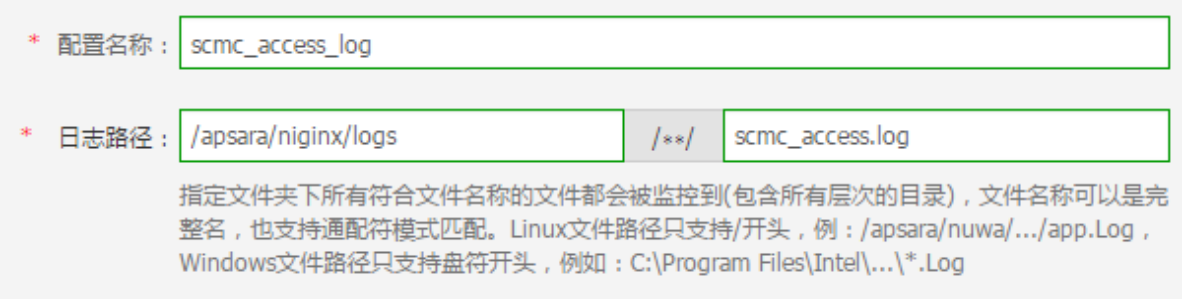
日志文件查找模式为多层目录匹配，即指定文件夹下所有符合文件名模式的文件都会被监控到，包含所有层次的目录。

- 例如`/apsara/nuwa/ ··· /*.log`表示`/apsara/nuwa`目录中（包含该目录的递归子目录）后缀名为`.log`的文件。
- 例如`/var/logs/app_* ··· /*.log*`表示`/var/logs`目录下所有符合`app_*`模式的目录中（包含该目录的递归子目录）文件名包含`.log`的文件。

说明：

一个文件只能被一个配置收集。

图 4-14: 指定目录和文件名



6. 设置收集模式。

Logtail支持极简模式、分隔符模式、JSON 模式、完整正则模式等方式收集日志，具体说明请参考采集方式。本示例以极简模式和完整正则模式为例介绍收集模式的设置。

- 极简模式

  目前极简模式即单行模式。单行模式下默认一行日志内容为一条日志，即日志文件中，以换行符分隔两条日志。单行模式下，不提取日志字段，即默认正则表达式为(.*)，同时记录当前服务器的系统时间作为日志产生的时间。如果后续您需要对极简模式进行更详细的设置，可以通过修改配置进入完整模式逐项调整。有关如何修改Logtail配置，参见管理采集配置。

  极简模式下，您只需要指定文件目录和文件名称，Logtail会按照每行一条日志进行收集，同时将日志时间设定为抓取该条日志时服务器的系统时间，不会提取日志内容中的字段。

- 完整正则模式

  如果需要对内容做更多个性化的字段提取设置（比如跨行日志，提取字段等），选择 完整正则模式 即可进行个性化定制。您可以参考 *Logtail*简介了解这些参数的具体含义和设置方式。

1. 输入 **日志样例**。

   让您提供日志样例的目的是方便日志服务控制台自动提取其中的正则匹配模式，请务必使用实际场景的日志。

2. 关闭 **单行模式**。

   默认为使用单行模式，即按照一行为一条日志进行分割，如果需要收集跨行日志（比如 Java 程序日志），需要关闭**单行模式**，然后设置 **行首正则表达式**。

3. 设置 **行首正则表达式**。

   提供**自动生成**和**手动输入**两种功能。填写完日志样例后，单击 **自动生成** 即会生成正则；如果无法自动生成，可以切换为手动模式输入进行验证。

4. 开启**提取字段**。

   如果需要对日志内容中的字段单独分析处理，可以使用 **提取字段** 功能将指定字段变成 Key-Value 对后发送到服务端，所以需要您指定解析一条日志内容的方式，即正则表达式。

   日志服务控制台提供两种方式让您指定解析正则表达式。第一种方式是通过简单交互自动生成正则表达式。您通过“划选”的方式操作日志样例，选中需要提取的字段，日志服务控制台会自动生成正则表达式。

   如下图所示：

图 **4-15:** 自动生成正则表达式



尽管自动生成方式避免了您自己写正则表达式的困扰，但是自动生成的正则表达式很多时候并不是完美的，您可以手动直接输入正则表达式。单击 **手动输入正则表达式** 切换到手动输入模式。手动输入完成后，单击右侧的**验证** 即会验证您输入的正则表达式是否可以解析、提取日志样例。

无论使用自动生成还是手动输入方式，产生日志解析正则表达式后，您都需要给每个提取字段命名，设定对应字段的 Key，如下图所示：

图 **4-16:**



5. 设置使用系统时间。

   默认设置 **手动输入正则表达式**。如果关闭，您需要在提取字段时指定某一字段（Value）为时间字段，并命名为 `time`（如上图）。在选取 `time` 字段后，您可以单击 **时间转换格式** 中的 **自动生成** 生成解析该时间字段的方式。关于日志时间格式的更多信息请参考配置时间格式。

7. （可选）配置高级选项，设置完成后，单击下一步。

   请根据您的需求配置 **本地缓存**、**配置日志*Topic*生成方式**、**日志文件编码**、**最大监控目录深度**、**超时属性**和**过滤器配置**。如没有特殊需求，可以保持默认配置。

配置项	详情
本地缓存	请选择是否打开**本地缓存**。当日志服务不可用时，日志可以缓存在机器本地目录，服务恢复后进行续传，默认最大缓存值1GB。
丢弃解析失败日志	请选择解析失败的日志是否上传到日志服务。 开启后，解析失败的日志不上传到日志服务；关闭后，日志解析失败时上传原始日志，其中Key为`__raw_log__`、Value为日志内容。

配置项	详情
上传原始日志	请选择是否需要上传原始日志。开启该功能后，原始日志内容会作为__raw__字段与解析过的日志一并上传。
Topic生成方式	• **空-不生成Topic**：默认选项，表示设置Topic为空字符串，在查询日志时不需要输入Topic即可查询。 • **机器组Topic属性**：设置Topic生成方式为机器组Topic属性，可以用于明确区分不同前端服务器产生的日志数据。 • **文件路径正则**：选择此项之后，您需要填写下方的**自定义正则**，用正则式从路径里提取一部分内容作为Topic。可以用于区分具体用户或实例产生的日志数据。
自定义正则	如您选择了**文件路径正则**方式生成Topic，需要在此处填写您的自定义正则式。
日志文件编码	• utf8：指定使用UTF-8编码。 • gbk：指定使用GBK编码。
最大监控目录深度	指定从日志源采集日志时，监控目录的最大深度，即最多监控几层日志。最大目录监控深度范围0-1000，0代表只监控本层目录。
超时属性	如果一个日志文件在指定时间内没有任何更新，则认为该文件已超时。您可以对**超时属性**指定以下配置。 • 永不超时：指定持续监控所有日志文件，永不超时。 • 30分钟超时：如日志文件在30分钟内没有更新，则认为已超时，并不再监控该文件。
过滤器配置	日志只有完全符合过滤器中的条件才会被收集。 例如： • **满足条件即收集**：配置`Key:level Regex:WARNING|ERROR`，表示只收集level为WARNING或ERROR类型的日志。 • 过滤不符合条件的数据： — 配置`Key:level Regex:^(?!.*(INFO|DEBUG))`，表示代表不收集level为INFO或DEBUG类型的日志。 — 配置`Key:url Regex:.*^(?!.*(healthcheck)).*`，表示不采集url中带有healthcheck的日志，例如key为url，value为`/inner/healthcheck/jiankong.html`的日志将不会被采集。 更多示例可参考*regex-exclude-word*、*regex-exclude-pattern*。

8. 勾选所需的机器组并单击**应用到机器组**将配置应用到机器组。

如果您还未创建机器组，需要先创建一个机器组。有关如何创建机器组，参见 创建IP地址机器组。

> 说明：
>
> - Logtail配置推送生效时间最长需要 3 分钟，请耐心等待。
> - 如果需要收集 IIS 的访问日志，请务必首先参考*Logstash 收集 IIS 日志*配置 IIS。
> - 创建Logtail配置后，您可以查看Logtail配置列表、修改Logtail配置或删除Logtail配置。详细信息，参见管理采集配置。

图 4-17: 应用到机器组



完成配置后，日志服务开始收集日志。

后续操作

完成以上操作后，您可以在页面指引下继续配置**查询分析 & 可视化**和**投递 & ETL**。

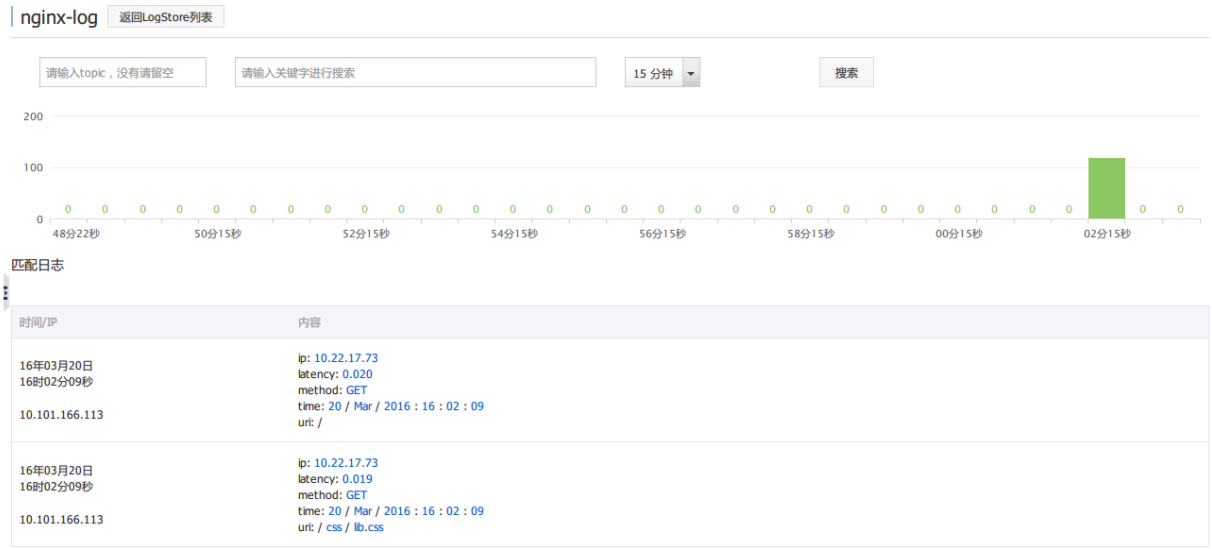
极简模式下收集到服务端的日志如下所示。每条日志的所有内容都在名为 **content** 的KEY下面。

图 **4-18:** 极简模式收集预览



完整正则模式下，收集到服务端的日志内容如下所示。每条日志的内容都按照设定的 Key-Value收集到了服务端。

图 **4-19:** 正则模式收集预览



Logtail配置项

配置Logtail时需要填写配置项，常用配置项具体描述与限制如下：

配置项	描述
日志路径	目录结构支持完整路径和通配符两种模式。通配符模式为多层目录匹配，即指定文件夹下所有符合文件名称的文件都会被监控到，包含所有层次的目录。
日志文件名	指定收集日志文件名称，区分大小写，可以使用通配符。例如`*.log`。Linux下的文件名通配符包括”*“, “?”和”[…]”。
本地存储	表示是否启用本地缓存临时存储因网络短暂中断而无法发送的日志。
日志首行头	指定多行日志的起始头，需指定正则表达式。在多行日志收集场景下（如应用程序日志中的堆栈信息），无法使用换行符来分割每条日志。这时需要指定一个多行日志的起始头，当发现该起始头则表示上条日志已经结束，新的一条已经开始。由于每条日志的起始头可能并不一样（如时间戳），故需要指定一个起始头的匹配规则，即这里的正则表达式。
日志解析表达式	定义如何提取一条日志信息，并转化成为日志服务日志的格式。用户需要指定一个正则表达式提

	取需要的日志字段信息，并且定义每个提取的字段名称。
日志时间格式	定义如何解析日志数据中的时间戳字符串的时间格式，具体请参见配置时间格式。

日志写入方式

除了使用Logtail收集日志外，日志服务还提供API和SDK的方式，以方便您写入日志。

- 使用 API 写入日志

  日志服务提供REST风格的API帮助您写入日志。您可以通过API中的 *PostLogStoreLogs* 接口写入数据。关于API的完整参考请见 概览。

- 使用 SDK 写入日志

  除了API，日志服务还提供了多种语言（Java、.NET、PHP 和 Python）的SDK方便您写入日志。关于SDK的完整参考请见 概述。

4.5.2 配置解析

指定日志行分割方式

一条完整的访问日志一般为一行一条，例如Nginx的访问日志，每条日志以换行符分割。例如以下两条访问日志：

```
10.1.1.1 - - [13/Mar/2016:10:00:10 +0800] "GET / HTTP/1.1" 0.011 180
404 570 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; 360se
)"
10.1.1.1 - - [13/Mar/2016:10:00:11 +0800] "GET / HTTP/1.1" 0.011 180
404 570 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; 360se
)"
```

Java应用中的程序日志，一条日志通常会跨越多行，因此只能通过日志开头的特征区分每条日志行首，例如以下 Java 程序日志：

```
[2016-03-18T14:16:16,000] [INFO] [SessionTracker] [SessionTrackerImpl.
java:148] Expiring sessions
0x152436b9a12aecf, 50000
0x152436b9a12aed2, 50000
0x152436b9a12aed1, 50000
0x152436b9a12aed0, 50000
```

以上Java日志起始字段均为时间格式，即行首正则表达式为：[\d+-\d+-\w+:\d+:\d+,\d+]\s.*。在控制台可按照如下格式填写：

图 **4-20:** 完整模式解析正则表达式



提取日志字段内容

根据日志服务数据模型要求，一条日志的内容包含一个或者多个Key-Value对，如果提取指定字段进行分析处理，需要设置正则表达式提取指定内容；如果不需要对日志内容进行处理，可以将整条日志做为一对Key-Value对。

对于上例中的访问日志，您可以选择提取字段或不提取字段。

- 提取字段

  正则表达式为`(\S+)\s-\s-\s\[(\S+)\s[^]]+]\s"(\w+).*`，提取内容为：`10.1.1.1`、`13/Mar/2016:10:00`和`GET`。

- 不提取字段

正则表达式为`(.*)`，提取内容为：`10.1.1.1 - - [13/Mar/2016:10:00:10 +0800] "GET / HTTP/1.1" 0.011 180 404 570 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; 360se)"`。

指定日志时间

根据日志服务数据模型要求，一条日志必须要有时间（time）字段，并且格式为UNIX时间戳。目前提供使用系统时间（即Logtail抓取该条日志的时间）或者日志内容中的时间做为日志的时间。

对于上例中的访问日志：

- 如果提取日志内容中的时间字段，时间为`13/Mar/2016:10:00:10`，时间表达式为`%d/%b/%Y:%H:%M:%S`。
- 抓取日志时的系统时间，时间为抓取日志时的时间戳。

4.5.3 配置时间格式

每条日志服务日志都必须包括该日志发生的时间戳信息，Logtail接入服务在采集用户日志文件中的日志数据时，必须提取该条日志中时间戳字符串并把它解析为时间戳。因此，Logtail需要您指定其日志的时间戳格式帮助解析。

Linux 平台下的 Logtail 支持 strftime 函数提供的所有时间格式。只需要您的日志时间戳字符串能够被该函数定义的日志格式所表达，即可以被 Logtail 解析并使用。

现实环境中的日志时间戳字符串格式非常多样化，为方便用户配置，Logtail 支持的常见日志时间格式如下：

支持格式	说明	示例
%a	星期的缩写。	Fri
%A	星期的全称。	Friday
%b	月份的缩写。	Jan
%B	月份的全称。	January
%d	每月第几天，十进制格式，范围为01~31。	07, 31
%h	月份的缩写，与%b相同。	Jan
%H	小时，24小时制。	22
%I	小时，12小时制。	11

支持格式	说明	示例
%m	月份，十进制格式。	08
%M	分钟，十时制格式，范围为00~59。	59
%n	换行符。	换行符
%p	本地的AM（上午）或PM（下午）。	AM/PM
%r	12小时制的时间组合，与`%I:%M:%S %p`相同。	11:59:59 AM
%R	小时和分钟组合，与`%H:%M`相同。	23:59
%S	秒数，十进制，范围为00~59。	59
%t	TAB符。	TAB符
%y	年份，十进制，不带世纪，范围为00~99。	04；98
%Y	年份，十进制。	2004；1998
%C	十进制世纪，范围为00~99。	16
%e	每月第几天，十进制格式，范围为1~31。如果是个位数字，前面需要加空格。	7，31
%j	一年天数的十进制表示，范围为00~366。	365
%u	星期的十进制表示，范围为1~7，1 表示周一。	2
%U	每年的第几周，星期天是一周的开始。范围为00~53。	23
%V	每年的第几周，星期一是一周的开始。如果一月份刚开始的一周>=4天，则认为是第1周，否则认为下一个星期一是第1周。范围为01~53。	24
%w	星期几，十进制格式 ，范围为0~6，0代表周日。	5

支持格式	说明	示例
%W	每年的第几周，星期一是一周的开始。范围为00~53。	23
%c	标准的日期、时间。	需要指定长日期、短日期等更多信息，可以考虑用上面支持的格式更精确表达。
%x	标准的日期。	需要指定长日期、短日期等更多信息，可以考虑用上面支持的格式更精确表达
%X	标准的时间。	需要指定长日期、短日期等更多信息，可以考虑用上面支持的格式更精确表达
%s	Unix时间戳。	1476187251

4.5.4 导入历史日志文件

Logtail默认只采集增量的日志文件，如果您需要导入历史文件，可使用Logtail自带的导入历史文件功能。

前提条件

- Logtail版本需在`0.16.6`及以上。
- 需要被采集的历史文件必须处于采集配置覆盖的范围下，且此前未被Logtail采集过。
- 历史文件的最后修改时间必须在配置Logtail之前。
- 本地事件导入最长延迟为1分钟。
- 由于加载本地配置属于特殊行为，Logtail会向服务器发送`LOAD_LOCAL_EVENT_ALARM`以提醒用户。

背景信息

Logtail基于事件进行文件采集，事件通常由监听或定期轮询文件修改产生。除以上方式外，Logtail还支持从本地文件中加载事件，以此驱动日志采集。历史文件采集就是基于本地事件加载实现的功能。

操作步骤

1. 创建采集配置。

   请根据采集文本日志创建采集配置并应用到机器组，确保文件处于对应采集配置覆盖范围。

2. 获取配置唯一标识。

   采集配置的唯一标识可在本地/usr/local/ilogtail/user_log_config.json 中获取，示例如下：

```
grep "##" /usr/local/ilogtail/user_log_config.json | awk '{print $1
}'
                    "##1.0##log-config-test$multi"
                    "##1.0##log-config-test$ecs-test"
                    "##1.0##log-config-test$metric_system_test"
                    "##1.0##log-config-test$redis-status"
```

3. 添加本地事件。

   本地事件保存的路径为 /usr/local/ilogtail/local_event.json，类型为标准JSON，格式为：

```
[
  {
    "config" : "${your_config_unique_id}",
    "dir" : "${your_log_dir}",
    "name" : "${your_log_file_name}"
   },
  {
    ...
    }
    ...
]
```

   - 配置项

配置项	说明	示例
config	步骤2获取的配置唯一标识。	##1.0#
dir	日志所在文件夹。 说明： 文件夹不能以/结尾。	/data
name	日志文件名，支持通配符。 说明： 支持通配符的Logtail版本在0.16.14及以上。	acces

   说明：

为防止Logtail加载无效的JSON，建议您将本地事件配置保存在临时文件中，编辑完成后拷贝到`/usr/local/ilogtail/local_event.json`文件中。

- 配置示例

```
$ cat /usr/local/ilogtail/local_event.json
[
  {
    "config": "##1.0##log-config-test$ecs-test",
    "dir": "/data/log",
    "name": "access.log*"
  },
  {
    "config": "##1.0##log-config-test$tmp-test",
    "dir": "/tmp",
    "name": "access.log.2017-08-09"
  }
]
```

- 检查**Logtail**是否加载配置

  通常情况下，本地`local_event.json`保存后，Logtail会在1分钟内将本地配置文件加载到内存中，并将`local_event.json`内容清空。

  您可以通过以下三种方式检查Logtail是否已经读取事件：

  - 若`local_event.json`文件被清空，说明Logtail已经读取到事件信息。
  - 检查`/usr/local/ilogtail/ilogtail.LOG`文件中是否包含`process local event`关键字。若`local_event.json`被清空但未查询到该组关键字，可能因为您的本地配置文件内容不合法而被过滤。
  - 通过查询诊断错误查询是否存在`LOAD_LOCAL_EVENT_ALARM`提示。

- 配置被加载但未采集到数据

  若Logtail已经加载配置但数据未被采集到，可能由以下几种原因造成：

  - 配置不合法。
  - 本地配置`config`不存在。
  - 日志文件不在Logtail采集配置已设定的路径下。
  - 该日志文件被Logtail采集过。

- 如何采集已经被采集过的数据

  如果您需要采集已经被采集过的数据，可通过以下方式实现：

  1. 执行命令`/etc/init.d/ilogtaild stop`停止Logtail。

2. 在`/tmp/logtail_check_point`文件中查找对应的日志文件路径。
3. 删除该日志文件的checkpoint（JSON object）并保存。
4. 按照本文步骤3添加本地事件。
5. 执行命令`/etc/init.d/ilogtaild start`启动Logtail。

4.5.5 生成主题



说明：

syslog不支持配置主题（Topic）。

Topic生成方式

用户可以在Logtail收集日志时设置Topic，也可以使用API/SDK上传数据时设置Topic。目前支持通过控制台设置Topic生成方式为空-不生成**Topic**、机器组**Topic**属性和文件路径正则。

- 空-不生成**Topic**

  通过控制台配置Logtail收集文本文件时，日志Topic生成方式默认为空-不生成**Topic**，即Topic为空字符串，在查询日志时不需要输入Topic即可查询。

- 机器组**Topic**属性

  机器组**Topic**属性方式用于明确区分不同服务器产生的日志数据。如果您的不同服务器日志数据均保存在相同文件路径或相同文件中，当您需要在收集日志时通过Topic区分不同服务器的日志数据，可以将机器分为不同的机器组，即在创建机器组时，为不同的机器组设置不同的**Topic**属性，并设置**Topic**生成方式为机器组**Topic**属性。将机器组应用之前创建的Logtail配置后，即完成对应配置。

  如选择机器组**Topic**属性，Logtail上报数据时会将机器所在机器组的Topic属性作为主题名称上传至日志服务，在查询日志时需要指定Topic，即需要指定目标机器组Topic属性为查询条件。

- 文件路径正则

  文件路径正则方式用于区分具体用户或实例产生的日志数据。如果服务日志根据不同的用户或者实例将日志记录在不同目录下面，但是只要下级目录、日志文相同件名称相同，日志服务在收集日志文件时就无法明确区分日志内容是由那个用户哪个产生的。此时可以设置**Topic**生成方式为文件路径正则，并且输入文件路径的正则表达式，配置Topic为实例名称。

当选择**文件路径正则**主题生成方式时，Logtail上报数据时会将实例名称作为主题名称上传至日志服务。根据您的目录结构和配置，会生成不同的Topic，在查询日志时需要指定主题名称为实例名称。

例如，在以下目录结构中，根据不同的用户或者实例将日志记录在不同目录下面。

```
/logs
  | - /userA/serviceA
    | - service.log
  | - /userB/serviceA
    | - service.log
  | - /userC/serviceA
    | - service.log
```

如果仅配置文件路径为*/logs*，文件名称为*service.log*，将三个service目录下的日志内容实时收集至服务端，但是无法明确区分日志内容具体由哪个用户或者实例产生。此时可以另外设置**Topic生成方式**为**文件路径正则**，并且输入正则表达式`/(.*)/serviceA/.*`提取实例名称。设置后会为不同目录下的日志生成不同的Topic，分别为“userA”、“userB”和“userC”，查询日志时可以指定Topic查询。

设置日志Topic

1. 根据采集文本日志，通过控制台配置Logtail。

   如您需要配置Topic生成方式为**机器组Topic属性**，请在创建机器组/修改机器组页面中配置**机器组Topic**。

2. 在Logtail配置页面中，展开**高级选项**，在**Topic生成方式**中选择Topic的生成方式。

图 4-21: 设置日志Topic



修改日志Topic

如您需要修改日志Topic的生成方式，请直接在Logtail配置界面修改**Topic生成方式**选项。

说明：

修改后的配置仅对生效后采集的新数据有效。

4.6 自定义插件

4.6.1 简介

日志服务Logtail除了支持采集文本日志及syslog之外，也通过Logtail插件支持配置若干数据源，例如Http、MySQL查询结果和MySQL Binlog。

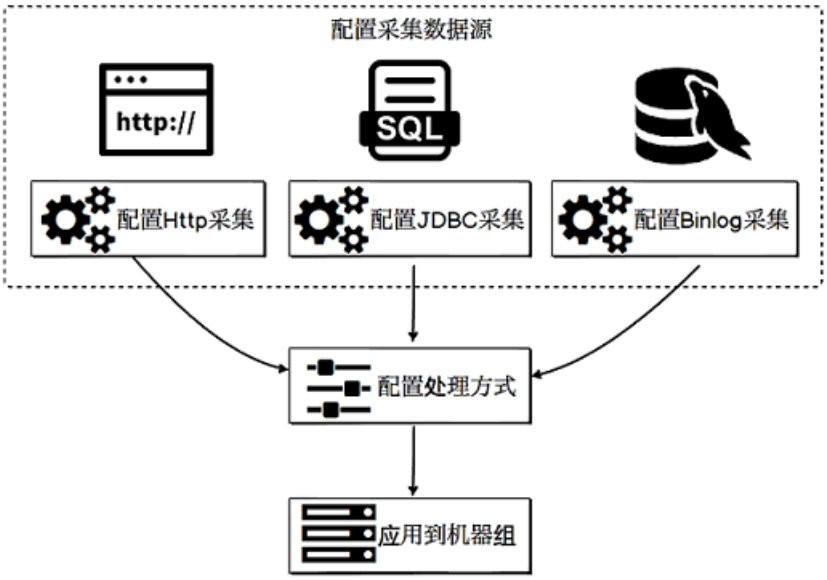
通过配置采集HTTP数据，并将处理结果上传到日志服务，可以进行实时的服务可用性检测和持续的可用性监控；配置MySQL查询结果为数据源，可以根据自增ID或时间等标志进行增量数据同步；配置SQL数据源以同步MySQL Binlog，可以增量订阅数据库改动，并进行实时查询与分析。

说明：

此功能目前仅支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。

配置流程

图 **4-22:** 配置流程



1. 配置输入源采集方式。

   每种不同的数据源具有不同的采集配置格式，请按照您的数据源类型选择对应的采集配置格式。

   - 采集 *MySQLBinlog*数据
   - 采集 *MySQL*查询结果
   - 采集 *HTTP*数据
   - 采集 容器标准输出

2. 配置处理方式。

   Logtail对于Binlog、MySQL查询结果、Nginx监控和HTTP输入源提供了统一的数据处理配置。用户可对一个输入源配置多个处理方式，各类输入源均支持所有类型的处理方式。Logtail会根据配置顺序逐一执行各个处理方式。

   详细说明请参考通用处理配置。

3. 应用到机器组。

   配置采集方式和处理方式之后，保存并应用到指定机器组，Logtail会自动拉取配置并开始采集。

4.6.2 MySQL Binlog方式

MySQL Binlog同步类似 *canal* 功能，以MySQL slave的形式基于Binlog进行同步，性能较高。

注意： 此功能目前只支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。

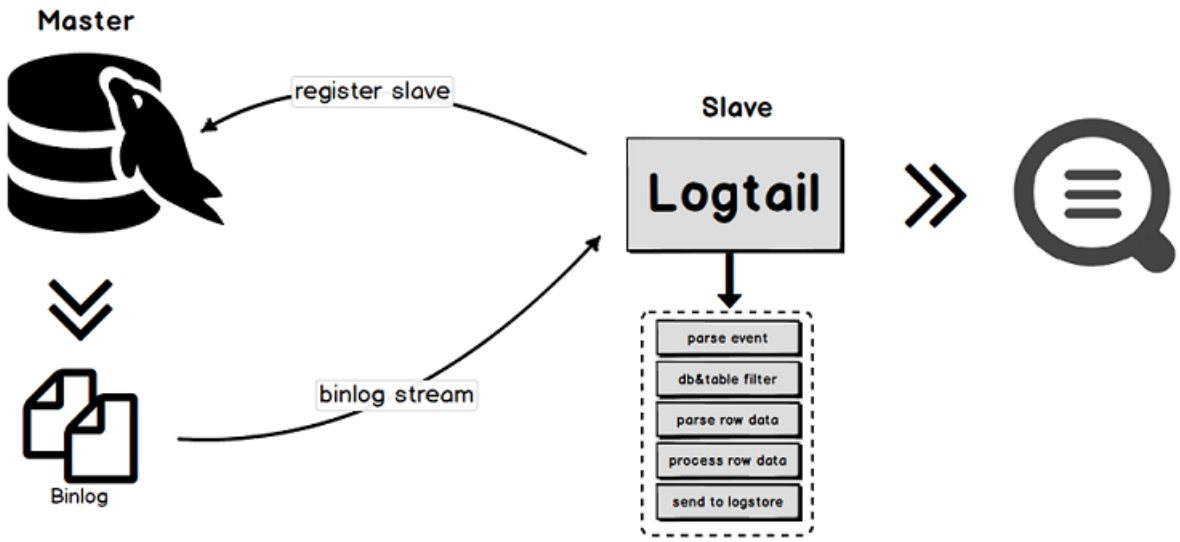
功能

- 通过Binlog订阅数据库增量更新数据，性能优越，支持RDS等MySQL协议的数据库。
- 支持数据库过滤（支持正则）。
- 支持同步点设置。
- 支持Checkpoint保存同步状态。

实现原理

如下图所示，Logtail内部实现了MySQL Slave的交互协议，将自己伪装成为MySQL的Slave节点，向MySQL master发送dump协议；MySQL的master收到dump请求后，会将自身的Binary log实时推送给Logtail，Logtail对Binlog进行事件解析、过滤、数据解析等，并将解析好的数据上传到日志服务。

图 **4-23:** 实现原理



应用场景

适用于数据量较大且性能要求较高的数据同步场景。

- 增量订阅数据库改动进行实时查询与分析。
- 数据库操作审计。

- 通过数据库更新信息使用日志服务进行自定义查询分析、可视化、对接下游流计算、导入MaxCompute离线计算、导入OSS长期存储等。

参数说明

MySQL Binlog方式输入源类型为：`service_canal`。

参数	类型	必选或可选	参数说明
Host	string	可选	数据库主机，默认为127.0.0.1。
Port	int	可选	数据库端口，默认为3306。
User	string	可选	数据库用户名，默认为root。
Password	string	可选	数据库密码；默认为空。
ServerID	int	可选	Logtail伪装成的Mysql Slave ID，默认为125。 说明：ServerID对于一个MySQL数据库必须唯一，否则同步失败。
IncludeTables	string 数组	可选	包含的表名称（包括db，例如`test_db.test_table`），为正则表达式，若某表不符合IncludeTables任一条件则该表不会被采集；不设置时默认收集所有表。 说明：若需要完全匹配，请在前后分别加上^$，例

参数	类型	必选或可选	参数说明
			如^test_db.test_table$。
ExcludeTables	string 数组	可选	忽略的表名称（包括db，例如test_db.test_table），为正则表达式，若某表符合ExcludeTables任一条件则该表不会被采集；不设置时默认收集所有表。 说明：若需要完全匹配，请在前后分别加上^$，例如^test_db.test_table$。
StartBinName	string	可选	首次采集的Binlog文件名，不设置时默认从当前时间点开始采集。
StartBinLogPos	int	可选	首次采集的Binlog文件名的offset，默认为0。
EnableGTID	bool	可选	是否附加全局事务ID，默认为true，为false时上传的数据将不附加。
EnableInsert	bool	可选	是否收集insert事件的数据，默认为true，为false时insert事件将不采集。
EnableUpdate	bool	可选	是否收集update事件的数据，默认为true，为false时update事件将不采集。

参数	类型	必选或可选	参数说明
EnableDelete	bool	可选	是否收集delete事件的数据，默认为true，为false时delete事件将不采集。
EnableDDL	bool	可选	是否收集DDL（data definition language）事件的数。 说明： 该选项默认为false，为false时DDL事件将不采集。该选项不支持`IncludeTables`、`ExcludeTables`过滤。
Charset	string	可选	编码方式，默认为`utf-8`。
TextToString	bool	可选	是否将`text`类型数据转换成string，默认为false。

使用限制

此功能目前仅支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。

- MySQL 必须开启Binlog，且Binlog必须为row模式（默认RDS已经开启）。

```
# 查看是否开启Binlog
mysql> show variables like "log_bin";
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | ON    |
+---------------+-------+
1 row in set (0.02 sec)
# 查看Binlog类型
mysql> show variables like "binlog_format";
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| binlog_format | ROW   |
+---------------+-------+
```

```
1 row in set (0.03 sec)
```

- ServerID 必须唯一，确保需同步的MySQL所有Slave的ID不重复。
- 需保证配置的用户具有需要采集的数据库读权限以及MySQL REPLICATION权限，示例如下：

```
CREATE USER canal IDENTIFIED BY 'canal';
GRANT SELECT, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'canal
'@'%';
-- GRANT ALL PRIVILEGES ON *.* TO 'canal'@'%' ;
FLUSH PRIVILEGES;
```

- 首次配置采集时如果不配置`StartBinName`，默认从当前时间点采集。

  如果想从指定位置采集，可以查看当前的Binlog以及offset，并将`StartBinName`、`StartBinLogPos`设置成对应的值，示例如下：

  说明：

  当指定StartBinName时，第一次采集会产生较大开销。

```
# StartBinName 设置成 "mysql-bin.000063", StartBinLogPos 设置成 0
mysql> show binary logs;
+------------------+-----------+
| Log_name         | File_size |
+------------------+-----------+
| mysql-bin.000063 |       241 |
| mysql-bin.000064 |       241 |
| mysql-bin.000065 |       241 |
| mysql-bin.000066 |     10778 |
+------------------+-----------+
4 rows in set (0.02 sec)
```

- 如安全需求较高，建议将SQL访问用户名和密码配置为xxx，待配置同步至本地机器后，在`/usr/local/ilogtail/user_log_config.json`文件找到对应配置进行修改。

  说明：

  - 修改完毕后请执行以下命令重启Logtail：

    ```
    sudo /etc/init.d/ilogtaild stop; sudo /etc/init.d/ilogtaild start
    ```

    。
  - 如果您再次在Web端修改此配置，您的手动修改会被覆盖，请再次手动修改本地配置。

- **RDS** 相关限制：

  — 无法直接在RDS服务器上安装Logtail，您需要将Logtail安装在能连通RDS实例的任意一台ECS上。

— RDS 只读备库当前不支持Binlog采集，您需要配置主库进行采集。

操作步骤

从RDS中同步`user_info`库中不以`_inner`结尾的表，配置如下。

1. 选择输入源。

   单击**数据接入向导**图标或**创建配置**，进入数据接入向导。并在**选择数据库类型**步骤中选择**BINLOG**。

2. 填写输入配置。

   进入**输入源配置**页面，填写**插件配置**。

   插件配置输入框中已为您提供配置模板，请根据您的需求替换配置参数信息。

   `inputs`部分为采集配置，是必选项；`processors`部分为处理配置，是可选项。采集配置部分需要按照您的数据源配置对应的采集语句，处理配置部分请参考处理采集数据配置一种或多种采集方式。

   说明：

   - 若安全需求较高，建议将SQL访问用户名/密码配置为xxx，待配置同步至本地机器后，在/usr/local/ilogtail/user_log_config.json文件找到对应配置进行修改。
   - 请将Host/User/Password/Port替换为实际访问参数。

   示例配置如下：

```
{
 "inputs": [
     {
         "type": "service_canal",
         "detail": {
             "Host": "************.mysql.rds.aliyuncs.com",
             "User" : "root",
             "ServerID" : 56321,
             "Password": "*******",
             "IncludeTables": [
                 "user_info\\..*"
             ],
             "ExcludeTables": [
                 ".*\\.\\S+_inner"
             ],
             "TextToString" : true,
             "EnableDDL" : true
         }
     }
 ]
```

```
}
```

3. 应用到机器组。

   进入**应用到机器组**页面。请在此处勾选**运行有此插件的Logtail机器组**。

   如您之前没有创建过机器组，单击**+创建机器组**以创建一个新的机器组。

   > 说明：
   > Binlog采集只需要一台安装Logtail的机器，您的机器组中只需要配置一个服务器IP即可。如您的机器组中有多台机器，请不要配置一样的ServerID。

4. 修改本地配置。

   如果您没有在**输入源配置**页面输入真实的URL、账号、密码等信息，需要在采集配置下发到本地后手动修改其中的内容。

   > 说明：
   > 如果您服务端输入的是真实信息，则无需此步骤。

   a. 登录Logtail所在服务器，查找`/usr/local/ilogtail/user_log_config.json`文件中`service_canal`关键词，修改下述对应的`Host`、`User`、`Password`等字段。

   b. 执行以下命令重启Logtail。

   ```
   sudo /etc/init.d/ilogtaild stop; sudo /etc/init.d/ilogtaild start
   ```

   Binlog采集配置已经完成，如果您的数据库有在进行对应的修改操作，则Logtail会立即将变化的数据采集到日志服务。

   > 说明：
   > Logtail默认采集Binlog的增量数据，如查看不到数据请确认在配置更新后数据库对应的表存在修改操作。

示例

例如，按照以上操作步骤配置处理方式后，对`user_info`下的`SpecialAlarm`表分别执行`INSERT`、`UPDATE`、`DELETE`操作。数据库表结构、数据库操作及Logtail采集的样例如下。

- 表结构

```
CREATE TABLE `SpecialAlarm` (
`id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`time` datetime NOT NULL,
```

```
`alarmtype` varchar(64) NOT NULL,
`ip` varchar(16) NOT NULL,
`count` int(11) unsigned NOT NULL,
PRIMARY KEY (`id`),
KEY `time` (`time`) USING BTREE,
KEY `alarmtype` (`alarmtype`) USING BTREE
) ENGINE=MyISAM AUTO_INCREMENT=1;
```

- 数据库操作

  对数据库执行INSERT、DELETE和UPDATE三种操作：

```
insert into specialalarm (`time`, `alarmType`, `ip`, `count`) values
(now(), "NO_ALARM", "10.10.**.***", 55);
delete from specialalarm where id = 4829235  ;
update specialalarm set ip = "10.11.***.**" where id = "4829234";
```

  并为`zc.specialalarm`创建一个索引：

```
ALTER TABLE `zc`.`specialalarm`
ADD INDEX `time_index` (`time` ASC);
```

- 样例日志

  在数据预览或在查找页面中，可以看到对应每种操作的样例日志如下：

  - INSERT语句

```
__source__:  10.30.**.**
__tag__:__hostname__:  iZbp145dd9fccu*****
__topic__:
_db_:  zc
_event_:  row_insert
_gtid_:  7d2ea78d-b631-11e7-8afb-00163e0eef52:536
_host_:  *********.mysql.rds.aliyuncs.com
_id_:  113
_table_:  specialalarm
alarmtype:  NO_ALARM
count:  55
id:  4829235
ip:  10.10.22.133
time:  2017-11-01 12:31:41
```

  - DELETE语句

```
__source__:  10.30.**.**
__tag__:__hostname__:  iZbp145dd9fccu****
__topic__:
_db_:  zc
_event_:  row_delete
_gtid_:  7d2ea78d-b631-11e7-8afb-00163e0eef52:537
_host_:  *********.mysql.rds.aliyuncs.com
_id_:  114
_table_:  specialalarm
alarmtype:  NO_ALARM
count:  55
id:  4829235
ip:  10.10.22.133
```

```
time:  2017-11-01 12:31:41
```

- UPDATE语句

```
__source__:  10.30.**.**
__tag__:__hostname__:  iZbp145dd9fccu****
__topic__:
_db_:  zc
_event_:  row_update
_gtid_:  7d2ea78d-b631-11e7-8afb-00163e0eef52:538
_host_:  *********.mysql.rds.aliyuncs.com
_id_:  115
_old_alarmtype:  NO_ALARM
_old_count:  55
_old_id:  4829234
_old_ip:  10.10.22.133
_old_time:  2017-10-31 12:04:54
_table_:  specialalarm
alarmtype:  NO_ALARM
count:  55
id:  4829234
ip:  10.11.145.98
time:  2017-10-31 12:04:54
```

- DDL（data definition language）语句

```
__source__:  10.30.**.**
__tag__:__hostname__:  iZbp145dd9fccu****
__topic__:
_db_:  zc
_event_:  row_update
_gtid_:  7d2ea78d-b631-11e7-8afb-00163e0eef52:539
_host_:  *********.mysql.rds.aliyuncs.com
ErrorCode:  0
ExecutionTime:  0
Query:  ALTER TABLE `zc`.`specialalarm`
ADD INDEX `time_index` (`time` ASC)
StatusVars:
```

元数据字段

binlog采集会将一些元数据和日志一起上传，具体上传的元数据列表如下：

字段	说明	示例
host	数据库host名称。	*********.mysql.rds.aliyuncs.com
_db_	数据库名称。	my-database
table	表的名称。	my-table
event	事件类型。	row_update、row_insert、row_delete等

字段	说明	示例
`_id_`	本次采集的自增ID，从0开始，每次采集一个binlog事件后加1。	`1`
`_gtid_`	GTID。	`7d2ea78d-b631-11e7-8afb-00163e0eef52:536`
`_filename_`	binlog文件名。	`binlog.001`
`_offset_`	binlog文件偏移量，该值只会在每次commit后更新。	`12876`

4.6.3 JDBC查询结果

以SQL形式定期采集数据库中的数据，根据SQL可实现各种自定义采集方式。



说明：

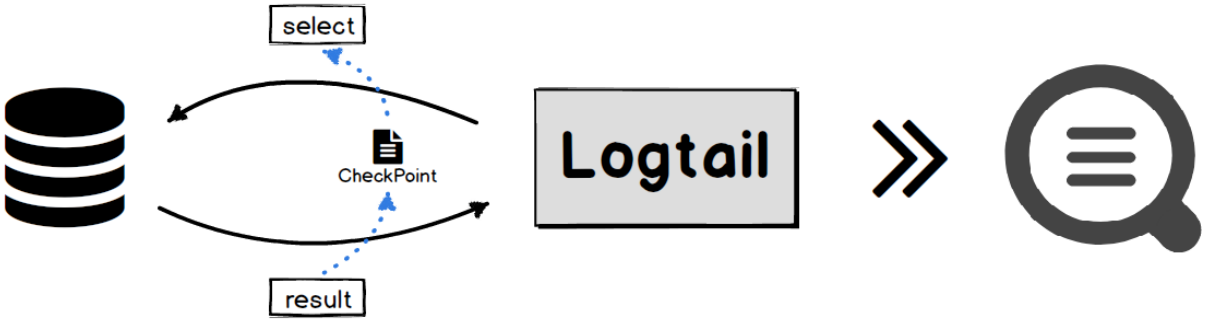
此功能目前仅支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。

功能

- 支持提供MySQL接口的数据库，包括RDS
- 支持分页设置
- 支持时区设置
- 支持超时设置
- 支持checkpoint状态保存
- 支持SSL
- 支持每次最大采集数量限制

实现原理

图 **4-24:** 实现原理



如上图所示，Logtail内部根据用户配置定期执行指定的SELECT语句，将SELECT返回的结果作为数据上传到日志服务。

Logtail在获取到执行结果时，会将结果中配置的CheckPoint字段保存在本地，当下次执行SQL时，会将上一次保存的CheckPoint带入到SELECT语句中，以此实现增量数据采集。

应用场景

- 根据自增ID或时间等标志进行增量数据同步。
- 自定义筛选同步。

参数说明

该输入源类型为：`service_mysql`。

参数	类型	必选或可选	参数说明
Address	string	可选	MySQL地址，默认为”127.0.0.1:3306”。
User	string	可选	数据库用户名，默认为”root”。
Password	string	可选	数据库密码，默认为空。
DialTimeOutMs	int	可选	数据库连接超时时间，单位为ms，默认为5000ms。

参数	类型	必选或可选	参数说明
ReadTimeOutMs	int	可选	数据库连接超时时间，单位为ms，默认为5000ms。
StateMent	string	必选	SQL语句。
Limit	bool	可选	是否使用Limit分页，默认为false。
PageSize	int	可选	分页大小，Limit为true时必须配置。
MaxSyncSize	int	可选	每次同步最大记录数，为0表示无限制，默认为0。
CheckPoint	bool	可选	是否使用checkpoint，默认为false。
CheckPointColumn	string	可选	checkpoint列名称，CheckPoint为true时必须配置。
CheckPoint ColumnType	string	可选	checkpoint列类型，支持`int`和`time`两种类型。
CheckPointStart	string	可选	checkpoint初始值。
CheckPoint SavePerPage	bool	可选	为true时每次分页时保存一次checkpoint，为false时每次同步完后保存checkpoint。
IntervalMs	int	必选	同步间隔，单位为ms。

使用限制

- 建议使用`Limit`分页，使用`Limit`分页时，SQL查询会自动在`StateMent`后追加LIMIT语句。
- 使用`CheckPoint`时，`StateMent`中SELECT出的数据中必须包含checkpoint列，且where条件中必须包含checkpoint列，该列的值填?

  例如checkpoint为”id”，`StateMent`为`SELECT * from ... where id > ?`。

- `CheckPoint`为true时必须配置`CheckPointColumn`、`CheckPointColumnType`、`CheckPointStart`。
- `CheckPointColumnType`只支持`int`和`time`类型，`int`类型内部存储为int64，`time`支持MySQL的date、datetime、time。

操作步骤

从MySQL中增量同步logtail.VersionOs中count > 0的数据，同步间隔为10s，checkpoint起始时间为2017-09-25 11:00:00，请求方式为分页请求，每页100，每次分页后保存checkpoint。具体配置步骤如下。

1. 选择输入源。

   单击**数据接入向导**图标或**创建配置**，进入数据接入向导。并在**选择数据库类型**步骤中选择**MYSQL查询结果**。

2. 填写输入配置。

   进入**输入源配置**页面，填写**插件配置**。

   插件配置输入框中已为您提供配置模板，请根据您的需求替换配置参数信息。

   `inputs`部分为采集配置，是必选项；`processors`部分为处理配置，是可选项。采集配置部分需要按照您的数据源配置对应的采集语句，处理配置部分请参考处理采集数据配置一种或多种采集方式。

   说明：

   若安全需求较高，建议将SQL访问用户名/密码配置为xxx，待配置同步至本地机器后，在/usr/local/ilogtail/user_log_config.json文件找到对应配置进行修改。

   示例配置如下：

```
{
 "inputs": [
     {
         "type": "service_mysql",
         "detail": {
             "Address": "********:3306",
             "User": "logtail",
             "Password": "********",
             "DataBase": "logtail",
             "Limit": true,
             "PageSize": 100,
             "StateMent": "SELECT * from logtail.VersionOs where
time > ?",
             "CheckPoint": true,
             "CheckPointColumn": "time",
```

```
            "CheckPointStart": "2017-09-25 11:00:00",
            "CheckPointSavePerPage": true,
            "CheckPointColumnType": "time",
            "IntervalMs": 10000
        }
    }
 ]
}
```

3. 应用到机器组。

   进入**应用到机器组**页面。请在此处勾选**支持此插件的Logtail机器组**。

   如您之前没有创建过机器组，单击**+创建机器组**以创建一个新的机器组。

4. 修改本地配置。

   如果您没有在**输入源配置**页面输入真实的URL、账号、密码等信息，需要在采集配置下发到本地后手动修改其中的内容。

   说明：

   如果您服务端输入的是真实信息，则无需此步骤。

   a. 登录Logtail所在服务器，查找`/usr/local/ilogtail/user_log_config.json`文件中`service_mysql`关键词，修改下述对应的`Address`、`User`、`Password`等字段。

   b. 执行以下命令重启Logtail。

   ```
   sudo /etc/init.d/ilogtaild stop; sudo /etc/init.d/ilogtaild start
   ```

示例

例如，按照以上操作步骤配置处理方式后，即可在日志服务控制台查看采集处理过的日志数据。表结构和Logtail采集的日志样例如下。

- 表结构

```
CREATE TABLE `VersionOs` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT COMMENT 'id',
  `time` datetime NOT NULL,
  `version` varchar(10) NOT NULL DEFAULT '',
  `os` varchar(10) NOT NULL,
  `count` int(11) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `timeindex` (`time`)
)
```

- 样例输出

```
"count":  "4"
"id:  "721097"
"os:  "Windows"
```

```
"time:  "2017-08-25 13:00:00"
"version":  "1.3.0"
```

4.6.4 HTTP方式

HTTP输入可根据您的配置定期请求指定URL，将请求返回body值作为输入源上传到日志服务。

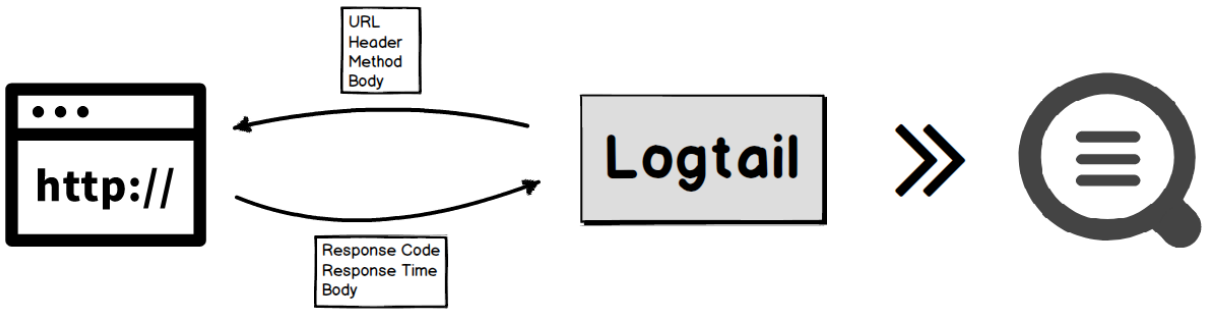
> 说明：
> 此功能目前仅支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。

功能

- 支持配置多个URL。
- 支持请求间隔配置。
- 支持HTTP方法配置。
- 支持自定义header。
- 支持设置method。
- 支持HTTPS。
- 支持检测body是否匹配固定pattern。

实现原理

图 **4-25:** 实现原理



如上图所示，Logtail内部会根据用户配置的HTTP请求url、method、header、body等信息定期对指定URL发起请求，将请求的返回的状态码、body内容以及响应时间做为数据上传到日志服务。

应用场景

- 应用状态监控（以HTTP方式提供监控接口），例如：
  - Nginx
  - Docker（以HTTP方式）

- Elastic Search
- Haproxy
- 其他以HTTP提供监控接口的服务

- 服务可用性检测，定期请求服务，通过状态码以及请求延迟做可用性监控。
- 数据定期拉取，例如微博评论、粉丝数等。

参数说明

参数说明

该输入源类型为：`metric_http`。

参数	类型	必选或可选	参数说明
Addresses	string 数组	必选	URL列表。 **说明：** 必须以`http`或`https`开头。
IntervalMs	int	必选	每次请求间隔，单位ms。
Method	string	可选	请求方法名，大写，默认为`GET`。
Body	string	可选	http body字段内容，默认为空。
Headers	key:string value:string map	可选	http header内容，默认为空。
PerAddressSleepMs	int	可选	Addresses列表中每个url请求间隔时间，单位ms，默认100ms。
ResponseTimeoutMs	int	可选	请求超时时间，单位ms，默认5000ms。
IncludeBody	bool	可选	是否采集请求的body，若为true，则将body以`content`为key存放，默认为false。

参数	类型	必选或可选	参数说明
FollowRedirects	bool	可选	是否自动处理重定向，默认为false。
InsecureSkipVerify	bool	可选	是否跳过https安全检查，默认为false。
ResponseStringMatch	string	可选	对于返回body进行正则表达式检查，检查结果以`_response_match_`为key存放：若匹配，value为`yes`；若不匹配，value为`false`。

使用限制

- 此功能目前仅支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。
- URL必须以`http`或`https`开头。
- 目前不支持自定义证书。
- 不支持交互式通信方式。

默认字段

每次请求默认会上传以下字段。

字段名	说明	示例
`_address_`	请求地址。	“*http://127.0.0.1/ngx_sta*
`_method_`	请求方法。	“GET”
`_response_time_ms_`	响应延迟，单位为ms。	“1.320”
`_http_response_code_`	状态码。	“200”
`_result_`	是否成功，取值范围:`success`、`invalid_body`、`match_regex_invalid`、`mismatch`、`timeout`。	“success”
`_response_match_`	返回body是否匹配`ResponseStringMatch`字段，若不存在`ResponseStringMatch`字段则为空，取值范围:`yes`、`no`。	“yes”

操作步骤

每隔1000ms请求一次nginx status模块，URL为 `http://127.0.0.1/ngx_status`，将返回的body使用正则提取出其中的状态信息。详细配置如下：

1. 选择输入源。

   单击**数据接入向导**图标或**创建配置**，进入数据接入向导。并在**选择数据库类型**步骤中选择**Logtail自定义插件**。

2. 填写输入配置。

   进入**输入源配置**页面，填写**插件配置**。

   `inputs`部分为采集配置，是必选项；`processors`部分为处理配置，是可选项。采集配置部分需要按照您的数据源配置对应的采集语句，处理配置部分请参考处理采集数据配置一种或多种采集方式。

   示例配置如下：

```
{
 "inputs": [
     {
         "type": "metric_http",
         "detail": {
             "IntervalMs": 1000,
             "Addresses": [
                 "http://127.0.0.1/ngx_status"
             ],
             "IncludeBody": true
         }
     }
 ],
 "processors" : [
     {
         "type": "processor_regex",
         "detail" : {
             "SourceKey": "content",
             "Regex": "Active connections: (\\d+)\\s+server accepts
 handled requests\\s+(\\d+)\\s+(\\d+)\\s+(\\d+)\\s+Reading: (\\d+)
Writing: (\\d+) Waiting: (\\d+).*",
             "Keys": [
                 "connection",
                 "accepts",
                 "handled",
                 "requests",
                 "reading",
                 "writing",
                 "waiting"
             ],
             "FullMatch": true,
             "NoKeyError": true,
             "NoMatchError": true,
             "KeepSource": false
         }
```

```
        }
    ]
}
```

3. 应用到机器组。

   进入应用到机器组页面。请在此处勾选支持此插件的**Logtail**机器组。

   如您之前没有创建过机器组，单击**+创建机器组**以创建一个新的机器组。

示例

按照以上操作步骤配置处理方式后，即可在日志服务控制台查看采集处理过的日志数据。

解析后上传的日志数据除包含正则解析后的数据外，还包括HTTP请求附加的method、address、time、code、result信息。

```
"Index" : "7"
"connection" : "1"
"accepts" : "6079"
"handled" : "6079"
"requests" : "11596"
"reading" : "0"
"writing" : "1"
"waiting" : "0"
"_method_" : "GET"
"_address_" : "http://127.0.0.1/ngx_status"
"_response_time_ms_" : "1.320"
"_http_response_code_" : "200"
"_result_" : "success"
```

4.6.5 Syslog输入源

Logtail支持通过自定义插件采集syslog。

简介

在Linux上，本地的syslog数据可以通过rsyslog等syslog agent转发到指定服务器IP地址和端口。为指定服务器添加Logtail配置之后，Logtail插件会以TCP或UDP协议接收转发过来的syslog数据。并且，插件能够将接收到的数据按照指定的syslog协议进行解析，提取日志中的facility、tag(program)、severity、content等字段。syslog协议支持*RFC3164*和*RFC5424*。

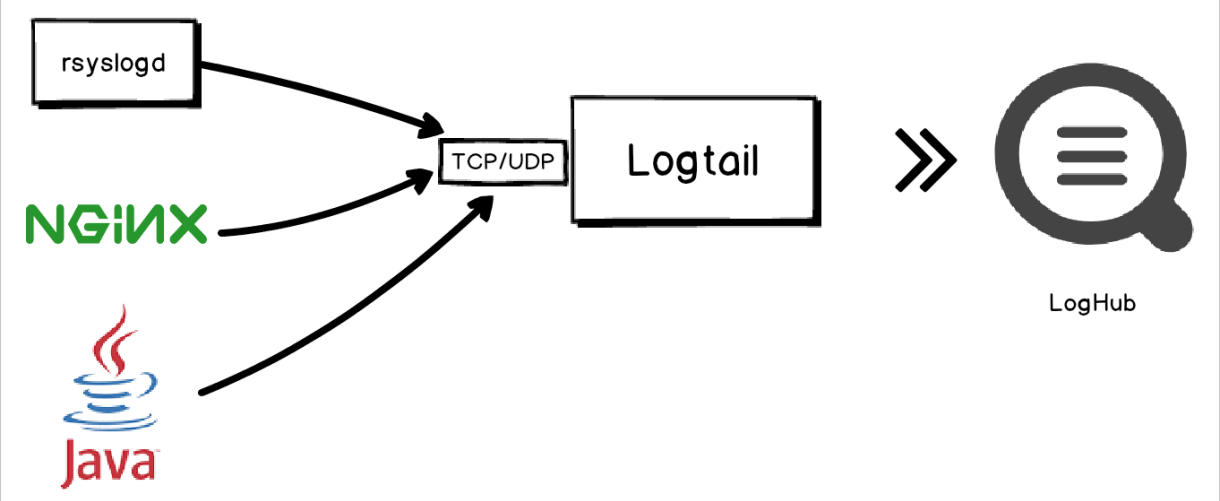


说明：

Windows Logtail不支持该插件。

实现原理

通过插件对指定的地址和端口进行监听后，Logtail能够作为syslog服务器采集来自各个数据源的日志，包括通过rsyslog采集的系统日志、 *Nginx*转发的访问日志或错误日志，以及*Java*等语言的syslog客户端库转发的日志。



注意事项

- Linux版 Logtail 0.16.13及以上版本支持该功能。
- Logtail可同时配置多个syslog插件，比如同时使用TCP和UDP监听127.0.0.1:9999。

Logtail配置项

该插件的输入类型为：`service_syslog`。

配置项	类型	是否必须	说明
Address	string	否	指定插件监听的协议、地址和端口，Logtail插件会根据配置进行监听并获取日志数据。格式为`[tcp/udp]://[ip]:[port]`，默认为`tcp://127.0.0.1:9999`。 **说明：** • Logtail插件配置监听的协议、地址和端口号必须与rsyslog配置文件设置的转发规则相同。 • 如果安装Logtail的服务器有多个IP地址可接收日志，可以将地址配置为0.0.0.0，表示监听服务器的所有IP地址。

配置项	类型	是否必须	说明
ParseProtocol	string	否	指定解析所使用的协议，默认为空，表示不解析。其中： • rfc3164：指定使用RFC3164协议解析日志。 • rfc5424：指定使用RFC5424协议解析日志。 • auto：指定插件根据日志内容自动选择合适的解析协议。
IgnoreParseFailure	boolean	否	指定解析失败后的行为，默认为true。其中： • true：放弃解析直接填充所返回的content字段。 • false：会丢弃日志。

默认字段

字段名	字段类型	字段含义
hostname	string	主机名，如果日志中未提供则获取当前主机名。
program	string	对应协议中的tag字段。
priority	string	对应协议中的priority字段。
facility	string	对应协议中的facility字段。
severity	string	对应协议中的severity字段。
unixtimestamp	string	日志对应的时间戳。
content	string	日志内容，如果解析失败的话，此字段包含未解析日志的所有内容。
_ip_	string	当前主机的IP地址。

前提条件

- 已创建Project和Logstore。
- 已创建机器组，并在机器组内服务器上安装了0.16.13及以上版本的Logtail。
- 需要被采集syslog的服务器上已安装了rsyslog。

配置Logtail插件采集syslog

1. 为rsyslog添加一条转发规则。

在需要采集syslog的服务器上修改rsyslog的配置文件*/etc/rsyslog.conf*，在配置文件的最后添加一行转发规则。添加转发规则后，rsyslog会将syslog转发至指定地址端口。

- 通过当前服务器采集本机syslog：配置转发地址为127.0.0.1，端口为任意非知名的空闲端口。
- 通过其他服务器采集本机syslog：配置转发地址为其他服务器的公网IP，端口为任意非知名的空闲端口。

例如以下配置表示将所有的日志都通过TCP转发至127.0.0.1:9000，配置文件详细说明请参考官网说明。

```
*.* @@127.0.0.1:9000
```

2. 执行以下命令重启rsyslog，使日志转发规则生效。

```
sudo service rsyslog restart
```

3. 登录日志服务控制台，单击Project名称。
4. 在**Logstore**列表页面单击**数据接入向导**图标。
5. 在**自定义数据**中单击**Logtail自定义插件**。
6. 填写**配置名称**。
7. 填写**插件配置**，并单击**下一步**。

`inputs`部分为采集配置，是必选项；`processors`部分为处理配置，是可选项。采集配置部分需要按照您的数据源配置对应的采集语句，处理配置部分请参考处理采集数据配置一种或多种采集方式。

同时监听UDP和TCP的示例配置如下：

```
{
    "inputs": [
        {
            "type": "service_syslog",
            "detail": {
                "Address": "tcp://127.0.0.1:9000",
                "ParseProtocol": "rfc3164"
            }
        },
        {
            "type": "service_syslog",
            "detail": {
                "Address": "udp://127.0.0.1:9001",
                "ParseProtocol": "rfc3164"
            }
        }
    ]
```

```
}
```



8. 应用到机器组。

   为机器组添加Logtail配置。请在此处勾选运行有此插件的**Logtail**机器组，并单击应用到机器组。

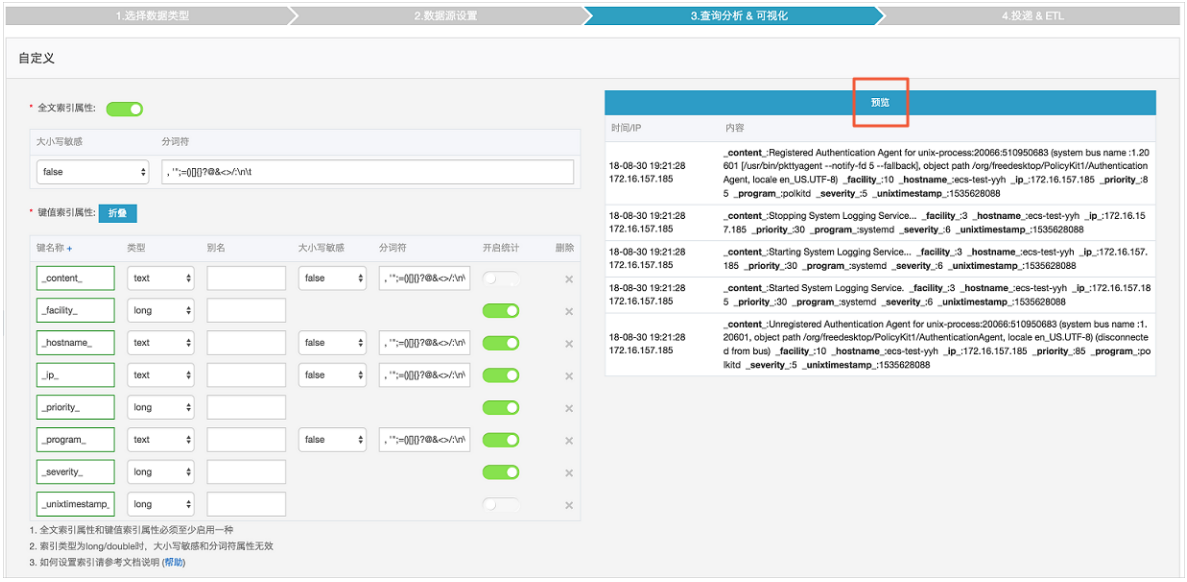
   > **说明：**
   > 接收syslog的服务器必须在机器组中，且安装了0.16.13及以上版本的Logtail。

   若您之前没有创建过机器组，单击**+创建机器组**创建一个新的机器组，再勾选机器组并单击**应用到机器组**。

9. 开启并配置索引。

   确保日志机器组心跳正常的情况下，可以点击右侧浏览按钮获取到采集上来的数据。

   如果您后续需要对采集到的syslog进行实时查询与分析，可以单击**展开**，根据日志内容和格式配置键值索引，后续可以按照指定字段进行查询和分析。



配置Logtail插件采集Nginx访问日志

Nginx支持直接把访问日志以syslog协议转发到指定地址和端口。如果您希望把服务器上包括Nginx访问日志在内的所有数据都以syslog的形式集中投递到日志服务，可以创建Logtail配置并应用到该服务器所在的机器组。

1. 在Nginx服务器的 *nginx.conf* 文件中增加转发规则。

   关于Nginx配置文件的详细信息请参考*Nginx*官网说明。

   例如，在配置文件中增加如下内容：

   ```
   http {
       ...

       # Add this line.
       access_log syslog:server=127.0.0.1:9000,facility=local7,tag=
   nginx,severity=info combined;

       ...
   }
   ```

2. 执行以下命令重启Nginx服务，使配置生效。

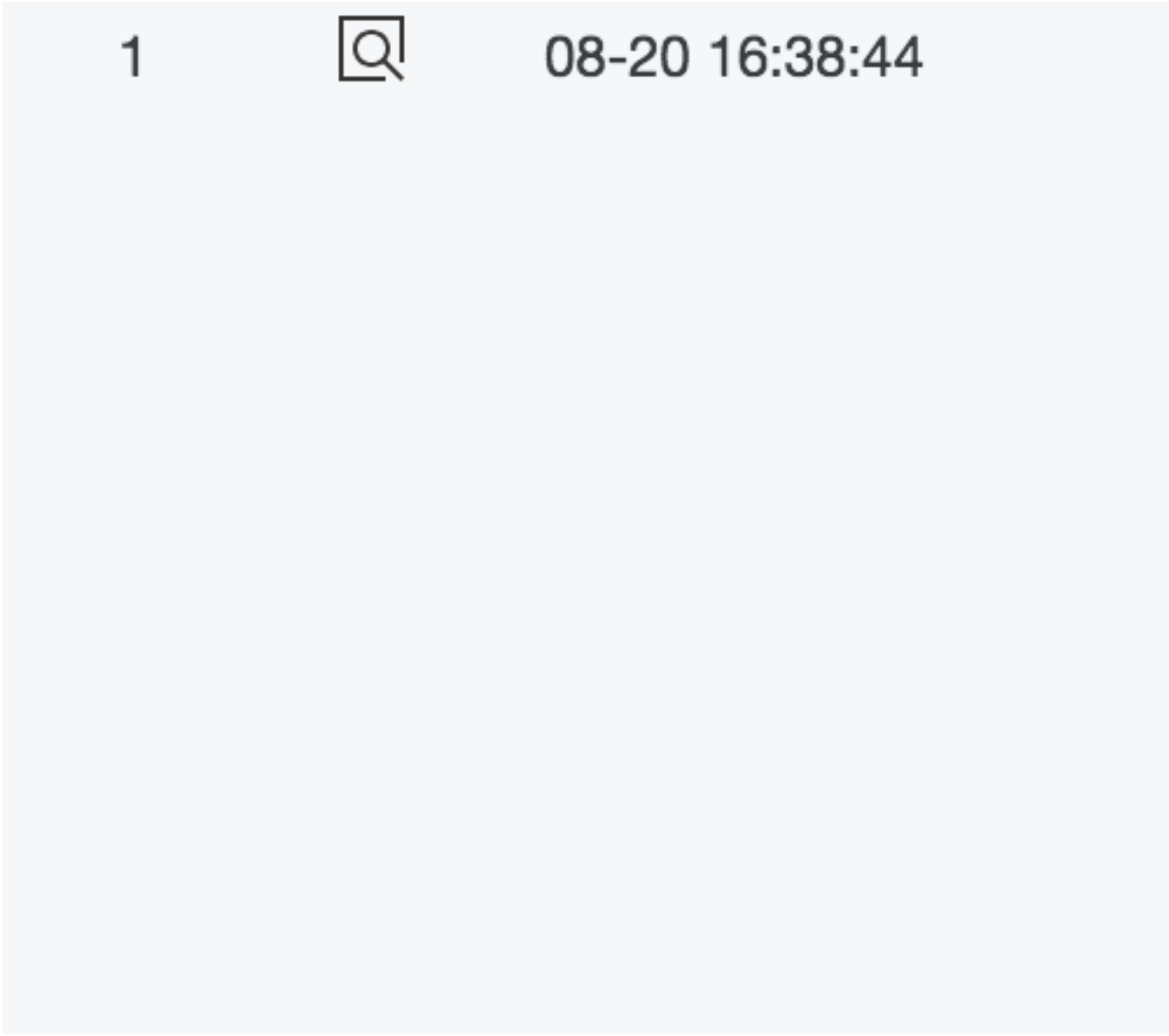
   ```
   sudo service nginx restart
   ```

3. 创建Logtail配置并应用到该服务器所在的机器组。

   配置过程请参考配置*Logtail*插件采集*syslog*。

4. 检验Logtail配置是否生效。

   在shell中执行命令`curl http://127.0.0.1/test.html`生成一条访问日志。如果采集配置已生效，可以在日志服务控制台的查询页面查看到日志信息。



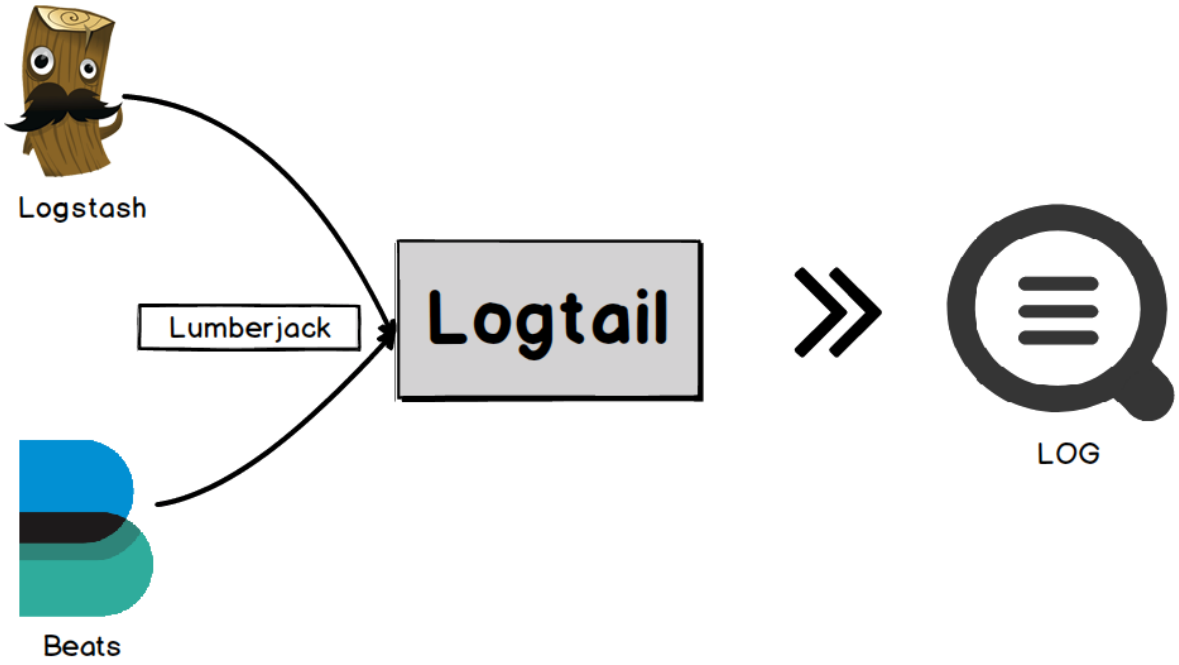
4.6.6 Beats和Logstash输入源

简介

Logtail除支持syslog协议外，还支持Lumberjack协议作为数据输入，可支持将Beats系列软件（MetricBeat、PacketBeat、Winlogbeat、Auditbeat、Filebeat、Heartbeat等）、Logstash采集的数据转发到日志服务。

实现原理

图 **4-26:** 实现原理



基于Logstash、Beats系列软件对Lumberjack协议的支持，日志服务可以通过Logtail，同时对Logstash、Beats系列软件进行Logtail Lumberjack插件监听配置，实现数据采集。

配置项

该输入源类型为: `service_lumberjack`。

> 说明：
> Logtail支持对采集的数据进行处理加工后上传，处理方式参见数据处理配置。

配置项	类型	是否必须	说明
BindAddress	string	否	绑定地址，默认为`127.0.0.1:5044`，IP和端口号均可自定义。如需被局域网内其他主机访问，请设置为0.0.0.0:5044。
V1	bool	否	Lumberjack V1版本协议版，默认为false。目

配置项	类型	是否必须	说明
			前Logstash支持的协议版本为V1
V2	bool	否	Lumberjack V2版本协议版，默认为true。目前Beats系列软件支持的协议版本均为V2。
SSLCA	string	否	证书授权机构（Certificate Authority）颁发的签名证书路径，默认为空。如果是自签名证书可不设置此选项。
SSLCert	string	否	证书的路径，默认为空。
SSLKey	string	否	证书对应私钥的路径，默认为空。
InsecureSkipVerify	bool	否	是否跳过SSL安全检查，默认为false。

使用限制

- 该功能在Logtail 0.16.9及以上版本支持。
- 如需将Logstash采集的数据上传，请参考*Logstash-Lumberjack-Output*。
- 如需将Beats（MetricBeat、PacketBeat、Winlogbeat、Auditbeat、Filebeat、Heartbeat等）系列软件采集的数据上传，请参考*Beats-Lumberjack-Output*。
- 同一Logtail可配置多个Lumberjack插件，但多个插件不能监听同一端口。
- 该插件支持SSL，Logstash采集的数据上传需要使用该功能

操作步骤

使用PacketBeat采集本地网络数据包，并使用Logtail Lumberjack插件上传到日志服务。详细配置如下：

1. 选择输入源。

   单击数据接入向导图标或创建配置，进入数据接入向导。并在选择数据库类型步骤中选择Logtail自定义插件。

2. 填写输入配置。

   进入输入源配置页面，填写插件配置。

   inputs部分为采集配置，是必选项；processors部分为处理配置，是可选项，由于Beats和Logstash输出的都是JSON格式数据，因此我们使用`processor_anchor`将json展开。

   关于处理配置部分请参考处理采集数据配置一种或多种采集方式。

```
{
  "inputs": [
    {
      "detail": {
        "BindAddress": "0.0.0.0:5044"
      },
      "type": "service_lumberjack"
    }
  ],
  "processors": [
    {
      "detail": {
        "Anchors": [
          {
            "ExpondJson": true,
            "FieldType": "json",
            "Start": "",
            "Stop": ""
          }
        ],
        "SourceKey": "content"
      },
      "type": "processor_anchor"
    }
  ]
}
```

3. 应用到机器组。

   进入应用到机器组页面。请在此处勾选支持此插件的Logtail机器组。

   如您之前没有创建过机器组，单击+创建机器组以创建一个新的机器组。

4. 配置PacketBeat。

   配置PacketBeat输出方式为`Logstash`，具体配置方式请参见*PacketBeat-Logstash-Output*。

   在本示例中。配置如下：

```
output.logstash:
```

```
  hosts: ["127.0.0.1:5044"]
```

示例

按照以上操作步骤配置处理方式后，可以尝试在本机输入命令`ping 127.0.0.1`，即可在日志服务控制台查看采集处理过的日志数据。

```
_@metadata_beat:  packetbeat
_@metadata_type:  doc
_@metadata_version:  6.2.4
_@timestamp:  2018-06-05T03:58:42.470Z
__source__:  **.**.**.**
__tag__:__hostname__:  *******
__topic__:
_beat_hostname:  bdbe0b8d53a4
_beat_name:  bdbe0b8d53a4
_beat_version:  6.2.4
_bytes_in:  56
_bytes_out:  56
_client_ip:  192.168.5.2
_icmp_request_code:  0
_icmp_request_message:  EchoRequest(0)
_icmp_request_type:  8
_icmp_response_code:  0
_icmp_response_message:  EchoReply(0)
_icmp_response_type:  0
_icmp_version:  4
_ip:  127.0.0.1
_path:  127.0.0.1
_responsetime:  0
_status:  OK
_type:  icmp
```

4.6.7 处理采集数据

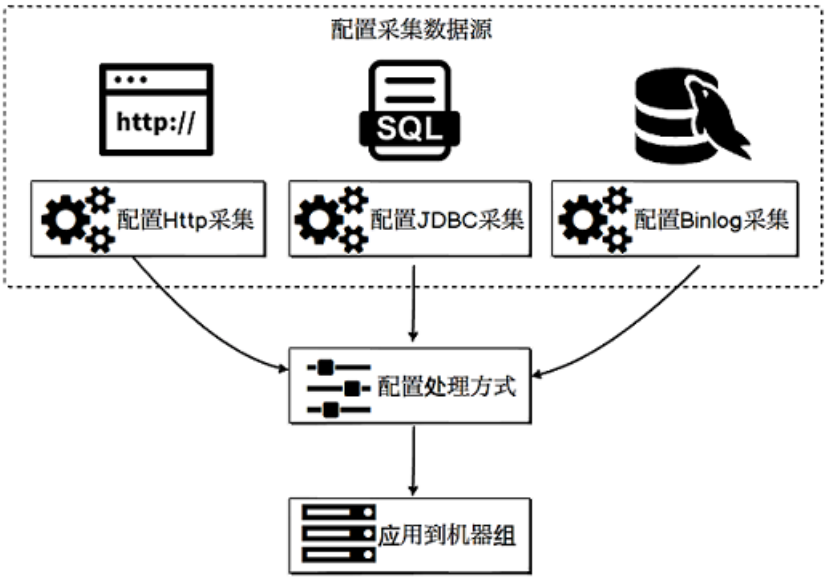
Logtail对于插件类型输入源提供了统一的数据处理配置，用户可对一个输入源配置多个处理方式，Logtail会根据配置顺序逐一执行各个处理方式。



说明：

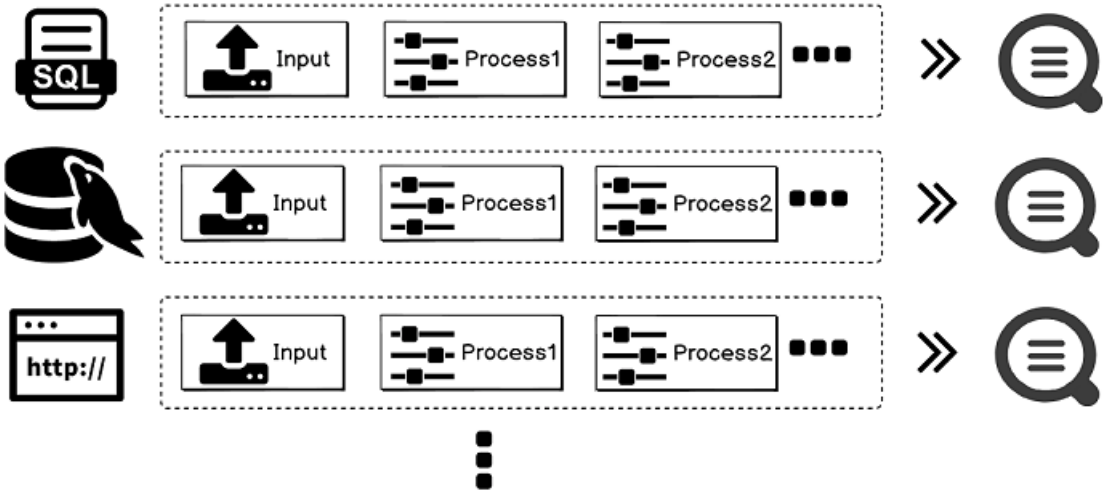
目前数据处理配置只支持插件类型输入源以及容器标准输出。

图 4-27: 采集过程



实现原理

图 4-28: 实现原理



处理方式

目前支持的处理方式如下：

- 正则提取
- 标定提取
- 单字分隔符
- 多字符分隔符

- GeoIP转换
- 正则过滤

您也可以根据以上处理方式，为您的输入源定制组合配置。

使用说明

为采集数据配置处理方式，处理配置的key为`processors`，value为json object的数组，数组内每个object代表一个处理方式配置。

单一处理方式包含两个字段:`type`、`detail`，其中`type`为该处理方式的类型，`detail`内部为该处理方式的详细配置。

```
"processors" : [
        {
            "type" : "processor_anchor",
            "detail" : {
                ...
            }
        },
        {
            "type" : "processor_regex",
            "detail" : {
                ...
            }
        }
    ]
```

正则提取

该方式通过对指定字段进行正则表达式匹配来提取其中匹配的字段。

参数说明

正则提取方式的类型（type）为`processor_regex`。

参数	类型	必选或可选	参数说明
SourceKey	string	必选	原始字段名，即需要进行正则提取的字段。
Regex	string	必选	用于匹配的正则表达式，需要提取的字段使用()标注，详细内容请参考维基百科。
Keys	string 数组	必选	需要提取的字段名，例如[“key1”, “key2” …]。

参数	类型	必选或可选	参数说明
NoKeyError	bool	可选	默认为false，为true时若没有找到SourceKey字段则报错。
NoMatchError	bool	可选	默认为false，为true时若正则不匹配则报错。
KeepSource	bool	可选	默认为false，为true时匹配完后不丢弃SourceKey字段。
FullMatch	bool	必选	默认为true，为true时只有字段完全匹配Regex时才会进行提取，为false时部分字段匹配也会进行提取。

示例

配置提取Access日志，详细配置如下：

- 输入

```
"content" : "10.200.**.** - - [10/Aug/2017:14:57:51 +0800] \"POST /
PutData?
Category=YunOsAccountOpLog&AccessKeyId=***************&Date=Fri
%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature
=*****************************
 HTTP/1.1\" 0.024 18204 200 37 \"-\" \"aliyun-sdk-java"
```

- 配置详情

```
{
    "type" : "processor_regex",
    "detail" : {"SourceKey" : "content",
         "Regex" : "([\\d\\.]+) \\S+ \\S+ \\[(\\S+) \\S+\\] \"(\\
w+) ([^\\\"]*)\" ([\\d\\.]+) (\\d+) (\\d+) (\\d+|-) \"([^\\\"]*)\"
 \"([^\\\"]*)\" (\\d+)",
         "Keys"   : ["ip", "time", "method", "url", "request_time",
 "request_length", "status", "length", "ref_url", "browser"],
         "NoKeyError" : true,
         "NoMatchError" : true,
         "KeepSource" : false
    }
```

```
}
```

- 处理后结果

```
"ip" : "10.200.**.**"
"time" : "10/Aug/2017:14:57:51"
"method" : "POST"
"url" : "/PutData?Category=YunOsAccountOpLog&AccessKeyId
=****************&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT
&Topic=raw&Signature=********************************"
"request_time" : "0.024"
"request_length" : "18204"
"status" : "200"
"length" : "27"
"ref_url" : "-"
"browser" : "aliyun-sdk-java"
```

标定提取

该方式通过标定指定字段起始和结束关键字进行提取，标定后的字符串支持直接提取和JSON展开方式。

参数说明

标定提取方式的类型（type）为`processor_anchor`。

参数	类型	必选或可选	参数说明
SourceKey	string	必选	原始字段名，即需要进行提取的字段。
Anchors	**Anchor** 数组	必选	标定项列表，具体参见下述表格。
NoAnchorError	bool	可选	默认为false，为true时若查找不到关键字则报错。
NoKeyError	bool	可选	默认为false，为true时若没有找到`SourceKey`字段则报错。
KeepSource	bool	可选	默认为false，为true时匹配完后不丢弃`SourceKey`字段。

Anchor类型说明

参数	类型	必选或可选	参数说明
Start	string	必选	起始关键字，若为空则代表匹配字符串开头。
Stop	string	必选	结束关键字，若为空则代表匹配字符串结尾。
FieldName	string	必选	提取的字段名。
FieldType	string	必选	提取字段类型，支持”string”、”json”两种类型。
ExpondJson	bool	可选	默认为false，为true且`FieldType`为`json`时将提取的json逐层展开。
ExpondConnecter	string	可选	JSON展开的连接符，默认为`_`。
MaxExpondDepth	int	可选	JSON展开最大深度，默认为`0`（无限制）。

示例

如下配置对某混合类型输入的处理结果如下：

- 输入

```
"content" : "time:2017.09.12 20:55:36\tjson:{\"key1\" : \"xx\", \"
key2\": false, \"key3\":123.456, \"key4\" : { \"inner1\" : 1, \"
inner2\" : false}}"
```

- 配置详情

```
{
   "type" : "processor_anchor",
   "detail" : {"SourceKey" : "content",
      "Anchors" : [
          {
              "Start" : "time",
              "Stop" : "\t",
              "FieldName" : "time",
              "FieldType" : "string",
              "ExpondJson" : false
          },
          {
              "Start" : "json:",
              "Stop" : "",
```

```
                "FieldName" : "val",
                "FieldType" : "json",
                "ExpondJson" : true
            }
        ]
    }
}
```

- 处理后结果

```
"time" : "2017.09.12 20:55:36"
"val_key1" : "xx"
"val_key2" : "false"
"val_key3" : "123.456"
"value_key4_inner1" : "1"
"value_key4_inner2" : "false"
```

单字符分隔符

该方式通过指定分隔符对字段进行分割，可指定Quote进行分隔符屏蔽。

参数说明

单字分隔符方式的类型（type）为`processor_split_char`。

参数	类型	必选或可选	参数说明
SourceKey	string	必选	原始字段名，即需要进行提取的字段。
SplitSep	string	必选	分隔符，必须为单字符，可以设置不可见字符，例如`\u0001`。
SplitKeys	string 数组	必选	切分后的字段名，例如[“key1”, “key2”…]。
QuoteFlag	bool	可选	默认为false，为true时代表使用quote。
Quote	string	可选	必须为单字符，QuoteFlag为true时生效，可以设置不可见字符，例如`\u0001`。
NoKeyError	bool	可选	默认为false，为true时若没有找到`SourceKey`字段则报错。

参数	类型	必选或可选	参数说明
NoMatchError	bool	可选	默认为false，为true时若切分。
KeepSource	bool	可选	默认为false，为true时匹配完后不丢弃SourceKey字段。

示例

对分隔符数据输入配置单字符分隔符处理方式，配置详情及处理结果如下：

- 输入

```
"content" : "10.**.**.**|10/Aug/2017:14:57:51 +0800|POST|PutData?
Category=YunOsAccountOpLog&AccessKeyId=***************&Date=Fri
%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature
=*************************|0.024|18204|200|37|-|
aliyun-sdk-java"
```

- 配置详情

```
{
   "type" : "processor_split_char",
   "detail" : {"SourceKey" : "content",
      "SplitSep" : "|",
      "SplitKeys" : ["ip", "time", "method", "url", "request_time",
 "request_length", "status", "length", "ref_url", "browser"]
  }
}
```

- 处理后结果

```
"ip" : "10.**.**.**"
"time" : "10/Aug/2017:14:57:51 +0800"
"method" : "POST"
"url" : "/PutData?Category=YunOsAccountOpLog&AccessKeyId
=***************&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT
&Topic=raw&Signature=********************************"
"request_time" : "0.024"
"request_length" : "18204"
"status" : "200"
"length" : "27"
"ref_url" : "-"
"browser" : "aliyun-sdk-java"
```

多字符分隔符

和单字符分隔符类似，多字符分隔符不支持Quote，完全按照分隔符拆分日志。

参数说明

多字符分隔符方式类型（type）为processor_split_string。

参数	类型	必选或可选	参数说明
SourceKey	string	必选	原始字段名，即需要进行提取的字段。
SplitSep	string	必选	分隔符，可以设置不可见字符，例如`\u0001\u0002`。
SplitKeys	string 数组	必选	切分后的字段名，例如[“key1”, “key2”…]。
PreserveOthers	bool	可选	默认为false, 为true时若分割的字段大于`SplitKeys`长度会保留超出部分。
ExpandOthers	bool	可选	默认为false，为true时继续解析超出部分。
ExpandKeyPrefix	string	可选	超出部分命名前缀，例如配置的`expand_`，则key为`expand_1`、`expand_2`。
NoKeyError	bool	可选	默认为false，为true时若没有找到`SourceKey`字段则报错。
NoMatchError	bool	可选	默认为false，为true时若切分。
KeepSource	bool	可选	默认为false，为true时匹配完后不丢弃`SourceKey`字段。

示例

对分隔符数据输入采用多字符分隔符方式处理，配置详情及处理结果如下：

- 输入

```
"content" : "10.**.**.**|#|10/Aug/2017:14:57:51 +0800|#|POST|#|
PutData?
Category=YunOsAccountOpLog&AccessKeyId=***************&Date=Fri
%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature
=******************************|#|0.024|#|18204|#|200|#|37|#|-|#|
```

```
aliyun-sdk-java"
```

- 配置详情

```
{
   "type" : "processor_split_string",
   "detail" : {"SourceKey" : "content",
      "SplitSep" : "|#|",
      "SplitKeys" : ["ip", "time", "method", "url", "request_time",
 "request_length", "status"],
      "PreserveOthers" : true,
      "ExpandOthers" : true,
      "ExpandKeyPrefix" : "expand_"
  }
}
```

- 处理后结果

```
"ip" : "10.**.**.**"
"time" : "10/Aug/2017:14:57:51 +0800"
"method" : "POST"
"url" : "/PutData?Category=YunOsAccountOpLog&AccessKeyId
=******************&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%
20GMT&Topic=raw&Signature=********************************"
"request_time" : "0.024"
"request_length" : "18204"
"status" : "200"
"expand_1" : "27"
"expand_2" : "-"
"expand_3" : "aliyun-sdk-java"
```

GeoIP转换

GeoIP转换对数据输入中的IP进行地理位置转换，能够将IP转换成：国家、省份、城市、经纬度。

 说明：

Logtail安装包本身并不带有GeoIP的数据库，需要您手动下载到本地并配置，建议下载精确到`City`的数据库。

参数说明

GeoIP转换插件类型（type）为`processor_geoip`。

参数	类型	必选或可选	参数说明
SourceKey	string	必选	原始字段名，即需要进行IP转换的字段。
DBPath	string	必选	GeoIP数据库的全路径，数据库格式为`mmdb`，例如`/user/data/GeoLite2`

参数	类型	必选或可选	参数说明
			`-City_20180102/GeoLite2-City.mmdb`。
NoKeyError	bool	可选	默认为false，为true时若没有找到`SourceKey`字段则报错。
NoMatchError	bool	可选	默认为false，为true时若IP地址无效或数据库中未匹配到该IP则上报错误。
KeepSource	bool	可选	默认为true，为false时转换完毕后丢弃`SourceKey`字段。
Language	string	可选	语言，默认为`zh-CN`，需确保您的GeoIP数据库中包含相应的语言。

示例

对输入的IP采用GeoIP转换成地理位置信息，配置详情及处理结果如下：

- 输入

```
"source_ip" : "**.**.**.**"
```

下载GeoIP数据库下载GeoIP数据库到安装Logtail的主机，例如可使用*MaxMind GeoLite2*中的*City*数据库



说明：

请检查数据库格式为mmdb类型。

- 配置详情

```
{
   "type": "processor_geoip",
    "detail": {
         "SourceKey": "ip",
         "NoKeyError": true,
         "NoMatchError": true,
```

```
        "KeepSource": true,
        "DBPath" : "/user/local/data/GeoLite2-City_20180102/
GeoLite2-City.mmdb"
    }
}
```

- 配置后结果

```
"source_ip_city_" : "**.**.**.**"
"source_ip_province_" : "浙江省"
"source_ip_city_" : "杭州"
"source_ip_province_code_" : "ZJ"
"source_ip_country_code_" : "CN"
"source_ip_longitude_" : "120.********"
"source_ip_latitude_" : "30.********"
```

正则过滤

该方式通过对字段进行正则表达式匹配过滤日志，可组合使用`Include`和`Exclude`两种方式。

参数说明

正则过滤方式的类型（type）为`processor_filter_regex`。

参数	类型	必选或可选	参数说明
Include	key:string value:string 的map	可选	key为日志字段，value为该字段匹配的正则表达式，若指定字段符合该表达式，则该条日志被收集。
Exclude	key:string value:string 的map	可选	key为日志字段，value为该字段匹配的正则表达式，若指定字段符合该表达式，则该条日志不被收集。



说明：

一条日志只有完全被`Include`中的参数匹配，且不被`Exclude`中的任一参数匹配时才会被采集，否则直接丢弃。

示例

对输入日志采正则过滤方式处理，配置详情及处理结果如下：

- 输入

— 日志1

```
"ip" : "10.**.**.**"
"method" : "POST"
...
"browser" : "aliyun-sdk-java"
```

— 日志2

```
"ip" : "10.**.**.**"
"method" : "POST"
...
"browser" : "chrome"
```

— 日志3

```
"ip" : "192.168.*.*"
"method" : "POST"
...
"browser" : "ali-sls-ilogtail"
```

- 配置详情

```
{
   "type" : "processor_filter_regex",
    "detail" : {
         "Include" : {
            "ip" : "10\\..*",
            "method" : "POST"
         },
         "Exclude" : {
            "browser" : "aliyun.*"
         }
    }
}
```

- 处理后结果

日志	是否匹配	原因
日志1	不匹配	browser匹配上了Exclude。
日志2	匹配	-
日志3	不匹配	Include中”ip”字段不匹配，不以”10”开头。

组合配置

各个处理配置可以组合搭配使用。您可以参考下述配置对日志先进行分隔符切分后，再对切分后的`detail`进行标定提取。

- 输入

```
"content" :
"ACCESS|QAS|11.**.**.**|1508729889935|52460dbed4d540b88a973cf5452b14
47|1238|appKey=ba,env=pub,requestTime=1508729889913,latency=22ms,
request={appKey:ba,optional:{\\domains\\:\\daily\\,\\version\\:\\v2\
\},rawQuery:{\\query\\:\\去乐山的路线\\,\\domain\\:\\导航\\,\\intent\\:
\\navigate\\,\\slots\\:\\to_geo:level3=乐山\\,\\location\\:\\北京\\},
requestId:52460dbed4d540b88a973cf5452b1447},
response={answers:[],status:SUCCESS}|"
```

- 配置详情

```
"processors" : [
      {
          "type" : "processor_split_char",
          "detail" : {"SourceKey" : "content",
              "SplitSep" : "|",
              "SplitKeys" : ["method", "type", "ip", "time", "req_id
", "size", "detail"]
          }
      },
      {
          "type" : "processor_anchor",
          "detail" : "SourceKey" : "detail",
              "Anchors" : [
                  {
                          "Start" : "appKey=",
                      "Stop" : ",env=",
                      "FieldName" : "appKey",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",env",
                      "Stop" : ",requestTime=",
                      "FieldName" : "env",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",requestTime=",
                      "Stop" : ",latency",
                      "FieldName" : "requestTime",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",latency=",
                      "Stop" : ",request=",
                      "FieldName" : "latency",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",request=",
                      "Stop" : ",response=",
                      "FieldName" : "request",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",response=",
                      "Stop" : "",
                      "FieldName" : "response",
```

```
                    "FieldType" : "json"
                }
            ]
        }
    }
]
```

- 处理后结果

```
"method" : "ACCESS"
"type" : "QAS"
"ip" : "**.**.**.**"
"time" : "1508729889935"
"req_id" : "52460dbed4d540b88a973cf5452b1447"
"size" : "1238"
"appKey" : "ba"
"env" : "pub"
"requestTime" : "1508729889913"
"latency" : "22ms"
"request" : "{appKey:nui-banma,optional:{\\domains\\:\\daily-faq\\,
\\version\\:\\v2\\},rawQuery:{\\query\\:\\\345\216\273\344\271\220\
345\261\261\347\232\204\350\267\257\347\272\277\\,\\domain\\:\\\345
\257\274\350\210\252\\,\\intent\\:\\navigate\\,\\slots\\:\\to_geo:
level3=\344\271\220\345\261\261\\,\\location\\:\\\345\214\227\344\
272\254\\},requestId:52460dbed4d540b88a973cf5452b1447}"
"response_answers" : "[]"
"response_status" : "SUCCESS"
```

4.7 容器日志采集

4.7.1 容器文本日志

Logtail支持将采集容器内产生的文本日志，并附加容器的相关元数据信息一起上传到日志服务。

功能特点

相对于基础的日志文件采集，Docker文件采集还具备以下功能特点：

- 只需配置容器内的日志路径，无需关心该路径到宿主机的映射
- 支持label指定采集的容器
- 支持label排除特定容器
- 支持environment指定采集的容器
- 支持environment指定排除的容器
- 支持多行日志（例如java stack日志等）
- 支持容器数据自动打标
- 支持Kubernetes容器自动打标

限制说明

- 采集停止策略。当container被停止后，Logtail监听到容器die的事件后会停止采集该容器日志的采集（延迟1~3秒），若此时采集出现延迟，则可能丢失停止前的部分日志。
- **Docker存储驱动限制**。目前只支持overlay、overlay2，其他存储驱动需将日志所在目录mount到本地。
- **Logtail运行方式**。必须以容器方式运行Logtail，且遵循Logtail部署方式进行部署。
- **Label**。 此处的label为docker inspect中的label信息，并不是Kubernetes配置中的label。
- **Environment**。此处的environment为容器启动中配置的environment信息。

配置流程

1. 部署并配置Logtail容器。
2. 设置服务端采集配置。

步骤1 Logtail部署和配置

- **Kubernetes**

  Kubernetes日志采集参见Logtail*Kubernetes*日志采集部署方案。

- 其他容器管理方式

  Swarm、Mesos等其他容器管理方式，请参考*Docker*日志采集通用部署方案。

步骤2 设置数据源

1. 在**Logstore**列表单击**数据接入向导**图标，进入配置流程。
2. 选择数据类型。

   单击**自建软件**中的**Docker文件**，并单击**下一步**。

3. 设置数据源。

配置项	是否必选	说明
是否为Docker文件	必须勾选	确认采集的目标文件是否为Docker文件。
Label白名单	可选	每项中LabelKey必填，若LabelValue不为空，则只采集容器label中包含LabelKey=LabelValue的容器；若LabelValue为空，则采集所有label中包含LabelKey的容器。

配置项	是否必选	说明
		说明： 1. 多个键值对间为或关系，即只要容器的label满足任一键值对即可被采集。 2. 此处的label为docker inspect中的label信息。
Label黑名单	可选	每项中LabelKey必填，若LabelValue不为空，则只排除容器label中包含LabelKey=LabelValue的容器；若LabelValue为空，则排除所有label中包含LabelKey的容器。 说明： 1. 多个键值对间为或关系，即只要容器的label满足任一键值对即可被排除。 2. 此处的label为docker inspect中的label信息。
环境变量白名单	可选	每项中EnvKey必填，若EnvValue不为空，则只采集容器环境变量中包含EnvKey=EnvValue的容器；若EnvValue为空，则采集所有环境变量中包含EnvKey的容器。 说明： • 多个键值对间为或关系，即只要容器的环境变量满足任一键值对即可被采集。 • 此处的environment为容器启动中配置的environment信息。
环境变量黑名单	可选	每项中EnvKey必填，若EnvValue不为空，则只排除容器环境变量中包含EnvKey=EnvValue的容器；若EnvValue为空，则排除所有环境变量中包含EnvKey的容器。 说明： • 多个键值对间为或关系，即只要容器的环境变量满足任一键值对即可被排除。 • 此处的environment为容器启动中配置的environment信息。

配置项	是否必选	说明
其他配置项	-	其他采集配置以及参数说明见采集文本日志。

说明：

- 本文中label白名单、黑名单与Kubernetes中定义的label不是同一概念，本文档中的label为Docker inspect中的label信息。
- Kubernetes中的namespace和容器名会映射到docker的label中，分别为`io.kubernetes.pod.namespace`和`io.kubernetes.container.name`。例如您创建的pod所属namespace为backend-prod，容器名为worker-server，则可以配置2个label白名单以指定只采集该容器的日志，分别为 `io.kubernetes.pod.namespace : backend-prod`和`io.kubernetes.container.name : worker-server`。
- Kubernetes中除`io.kubernetes.pod.namespace`和`io.kubernetes.container.name`外不建议使用其他label。其他情况请使用环境变量白名单或黑名单。

4. 应用到机器组。

   进入应用到机器组页面。请在此处勾选需要采集的Logtail机器组，并单击右下角的应用到机器组。如您之前没有创建过机器组，单击**+创建机器组**以创建一个新的机器组。

5. 完成容器文本日志接入流程。

   如您需要开启检索分析、投递配置等功能，请按照页面提示继续配置。

配置示例

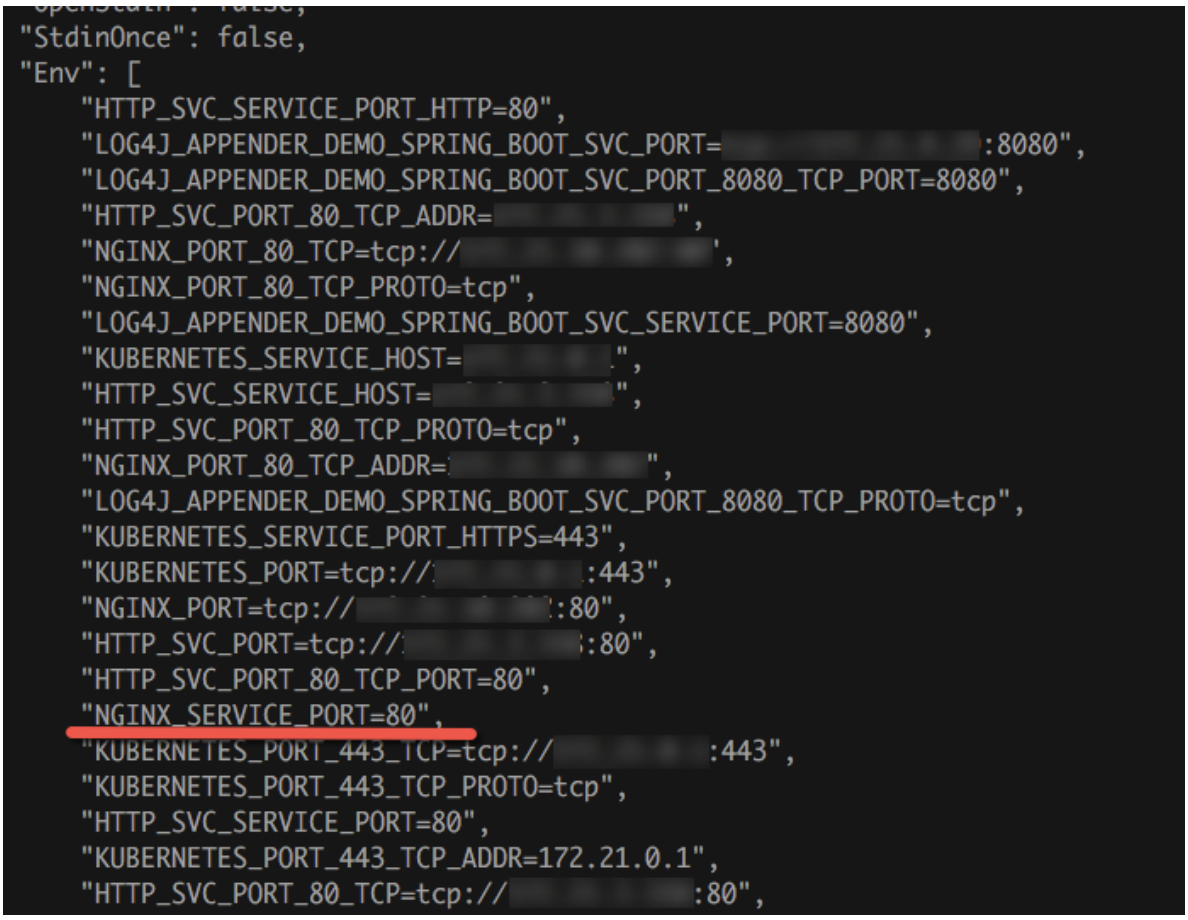
- **environment 配置方式**

  采集environment为`NGINX_PORT_80_TCP_PORT=80`且environment不为`POD_NAMESPACE=kube-system`的容器日志，日志文件路径为`/var/log/nginx/access.log`，日志解析方式为极简类型。

  说明：

  此处的environment为容器启动中配置的environment信息。

图 **4-29: environment** 配置方式示例



本示例中数据源配置如下。其他采集配置以及参数说明见采集文本日志。

图 4-30: 数据源配置示例



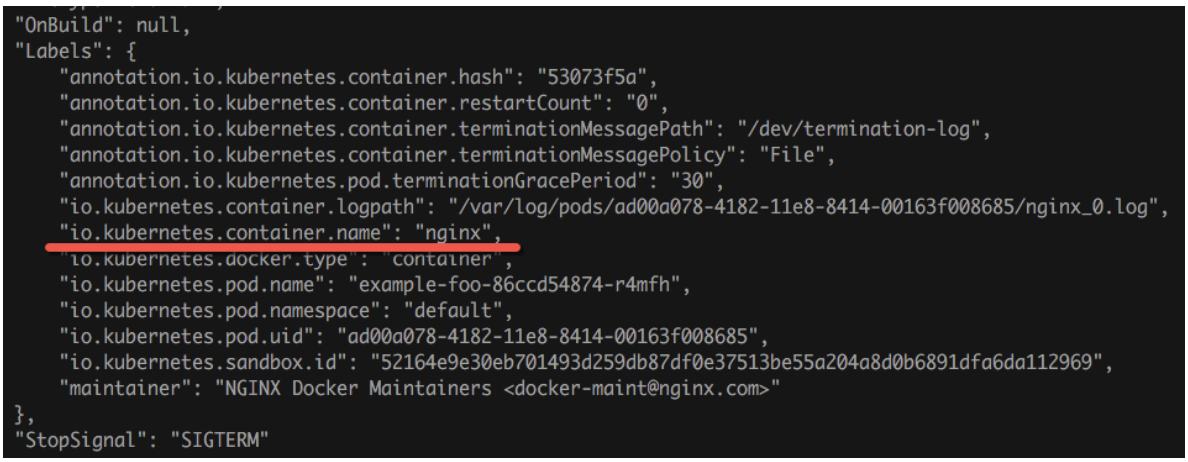
- **label 配置方式**

  采集label为`io.kubernetes.container.name=nginx`的容器日志，日志文件路径为`/var/log/nginx/access.log`，日志解析方式为极简类型。

  说明：

  此处的label为docker inspect中的label信息，并不是Kubernetes配置中的label。

图 **4-31: label**方式示例



本示例中数据源配置如下。其他采集配置以及参数说明见采集文本日志。

图 4-32: 数据源配置



默认字段

普通Docker

每条日志默认上传以下字段：

字段名	说明
_image_name_	镜像名
_container_name_	容器名

字段名	说明
_container_ip_	容器IP地址

Kubernetes

若该集群为Kubernetes，则默认每条日志上传以下字段：

字段名	说明
_image_name_	镜像名
_container_name_	容器名
_pod_name_	pod名
namespace	pod所在命名空间
_pod_uid_	pod的唯一标识
_container_ip_	pod的IP地址

4.7.2 容器标准输出

Logtail支持将容器的标准输出流作为输入源，并附加容器的相关元数据信息一起上传到日志服务。

功能特点

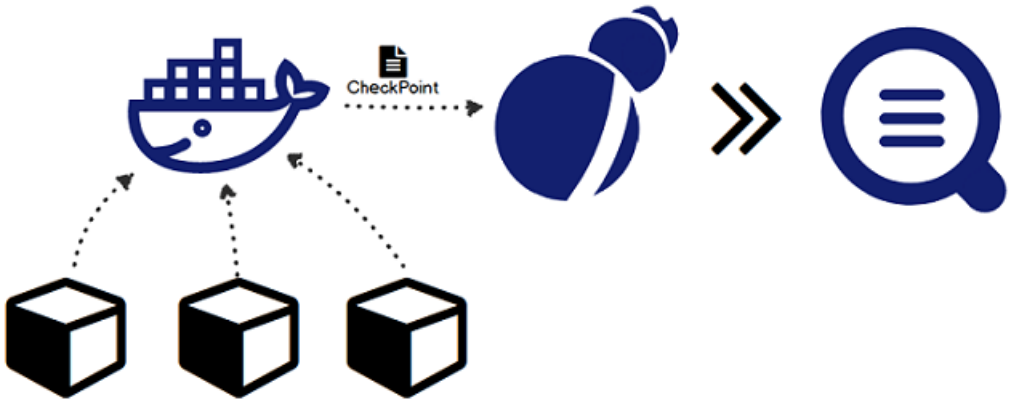
- 支持采集stdout、stderr
- 支持label指定采集的容器
- 支持label排除特定容器
- 支持environment指定采集的容器
- 支持environment指定排除的容器
- 支持多行日志（例如java stack日志等）
- 支持container数据自动打标
- 支持Kubernetes容器自动打标

实现原理

如下图所示，Logtail会与Docker的Domain Socket进行通信，查询该Docker上运行的所有container，并根据container配置的label和environment信息定位需要被采集的container。随后Logtail会通过docker的log命令获取指定container的日志。

Logtail在采集容器的标准输出时，会定期将采集的点位信息保存到checkpoint文件中，若Logtail停止后再次启动，会从上一次保存的点位开始采集日志。

图 **4-33:** 实现原理



使用限制

- 此功能目前仅支持Linux，依赖Logtail 0.16.0及以上版本，版本查看与升级参见*Linux*。
- Logtail默认通过`/var/run/docker.sock`访问docker engine，请确保该Domain Socket存在且具备访问权限。
- **多行日志限制**。为保证多行组成的一条日志不因为输出延迟而被分割成多条，多行日志情况下，采集的最后一条日志默认都会缓存一段时间。默认缓存时间为3秒，可通过`BeginLineTimeoutMs`设置，但此值不能低于1000，否则容易出现误判。
- **采集停止策略**。当container被停止后，Logtail监听到容器`die`的事件后会停止采集该container的标准输出，若此时采集出现延迟，则可能丢失停止前的部分输出。
- **Docker日志驱动类型限制**。目前标准输出采集仅支持JSON类型的日志驱动。
- **上下文限制**。默认一个采集配置在同一上下文中，若需要每种类型的container一个上下文，请单独配置。
- **数据处理**。采集到的数据默认字段为`content`，支持通用的处理配置。请参考简介配置一种或多种采集方式。
- **Label**。 此处的label为docker inspect中的label信息，并不是Kubernetes配置中的label。
- **Environment**。此处的environment为容器启动中配置的environment信息。

配置流程

1. 部署并配置Logtail容器。
2. 设置服务端采集配置。

步骤1 Logtail部署和配置

- **Kubernetes**

  Kubernetes日志采集参见Logtail *Kubernetes*日志采集部署方案。

- 其他容器管理方式

  Swarm、Mesos等其他容器管理方式，请参考*Docker*日志采集通用部署方案。

步骤2 设置数据源

1. 在**Logstore**列表单击数据接入向导图标，进入配置流程。
2. 选择数据类型。

   单击自建软件中的**Docker**标准输出，并单击下一步。
3. 设置数据源。

   在输入源配置页面，填写您的采集配置。示例如下，配置项说明请查看本文档中配置项说明部分。

```
{
 "inputs": [
     {
         "type": "service_docker_stdout",
         "detail": {
             "Stdout": true,
             "Stderr": true,
             "IncludeLabel": {
                 "io.kubernetes.container.name": "nginx"
             },
             "ExcludeLabel": {
                 "io.kubernetes.container.name": "nginx-ingress-
controller"
             },
             "IncludeEnv": {
                 "NGINX_SERVICE_PORT": "80"
             },
             "ExcludeEnv": {
                 "POD_NAMESPACE": "kube-system"
             }
         }
     }
 ]
}
```

4. 应用到机器组。

   进入应用到机器组页面。请在此处勾选需要采集的Logtail机器组。如您之前没有创建过机器组，单击**+创建机器组**以创建一个新的机器组。

配置项说明

该输入源类型为: `service_docker_stdout`。

> 说明：
> Logtail支持对采集的数据进行处理加工后上传，处理方式参见处理采集数据。

配置项	类型	是否必选	说明
IncludeLabel	map类型，其中key为string，value为string	必选	默认为空，为空时代表采集所有Container数据；当key非空，value为空时，代表包含label中所有包含此key的container。 说明： 1. 多个键值对间为或关系，即只要容器的label满足任一键值对即可被采集。 2. 此处的label为docker inspect中的label信息。
ExcludeLabel	map类型，其中key为string，value为string	可选	默认为空，为空时不排除任何Container；当key非空，value为空时，代表排除label中所有包含此key的container。 说明： 1. 多个键值对间为或关系，即只要容器的label满足任一键值对即被排除。 2. 此处的label为docker inspect中的label信息。
IncludeEnv	map类型，其中key为string，value为string	可选	默认为空，为空时代表采集所有Container数据；当key非空，value为空时，代表包含environment中所有包含此key的container。 说明： 1. 多个键值对间为或关系，即只要容器的environment满足任一键值对即可被采集。 2. 此处的environment为容器启动中配置的environment信息。
ExcludeEnv	map类型，其中key为string，value为string	可选	默认为空，为空时不排除任何Container；当key非空，value为空时，代表排除Environment中所有包含此key的container。

配置项	类型	是否必选	说明
			说明： 1. 多个键值对间为或关系，即只要容器的environment满足任一键值对即被排除。 2. 此处的environment为容器启动中配置的environment信息。
Stdout	bool	可选	默认为true，为false时不采集stdout数据。
Stderr	bool	可选	默认为true，为false时不采集stderr数据。
BeginLineRegex	string	可选	默认为空，非空时为行首匹配的正则表达式。若该表达式匹配某行，则将该行作为新的一条日志；否则将此行数据连接到上一条日志。
BeginLineTimeoutMs	int	可选	行首匹配超时时间，默认为3000，单位为毫秒。若3秒内没有新日志出现，则将最后一条日志输出。
BeginLineCheckLength	int	可选	行首匹配的长度，默认为10×1024，单位为字节。若行首规则在前N个字节即可体现，可设置此参数，以此提升行首匹配效率。
MaxLogSize	int	可选	日志最大长度，默认为512×1024，单位为字节。若日志超过该项配置，则不继续查找行首，直接上传。

说明：

1. 本文档中IncludeLabel、ExcludeLabel和Kubernetes中定义的label不是同一概念，本文档中的label为docker inspect中的label信息。
2. Kubernetes中的namespace和容器名会映射到docker的label中，分别为`io.kubernetes.pod.namespace`和`io.kubernetes.container.name`。例如您创建的pod所属namespace为backend-prod，容器名为 worker-server，则IncludeLabel中可以配置2个键值对来指定只采集该容器的日志，分别为 `io.kubernetes.pod.namespace : backend-prod`和`io.kubernetes.container.name : worker-server`。
3. Kubernetes不建议使用除`io.kubernetes.pod.namespace`和`io.kubernetes.container.name`之外的其他label。其他情况请使用IncludeEnv/ExcludeEnv。

默认字段

普通Docker

每条日志默认上传以下字段：

字段名	说明
time	数据时间，样例为2018-02-02T02:18:41.979147844Z
source	输入源类型，stdout 或 stderr
_image_name_	镜像名
_container_name_	容器名
_container_ip_	容器IP

Kubernetes

若该集群为Kubernetes，则默认每条日志上传以下字段：

字段名	说明
time	数据时间，样例为2018-02-02T02:18:41.979147844Z
source	输入源类型，stdout 或 stderr
_image_name_	镜像名
_container_name_	容器名
_pod_name_	pod名
namespace	pod所在命名空间
_pod_uid_	pod的唯一标识
_container_id_	pod的IP地址

配置示例

常规配置

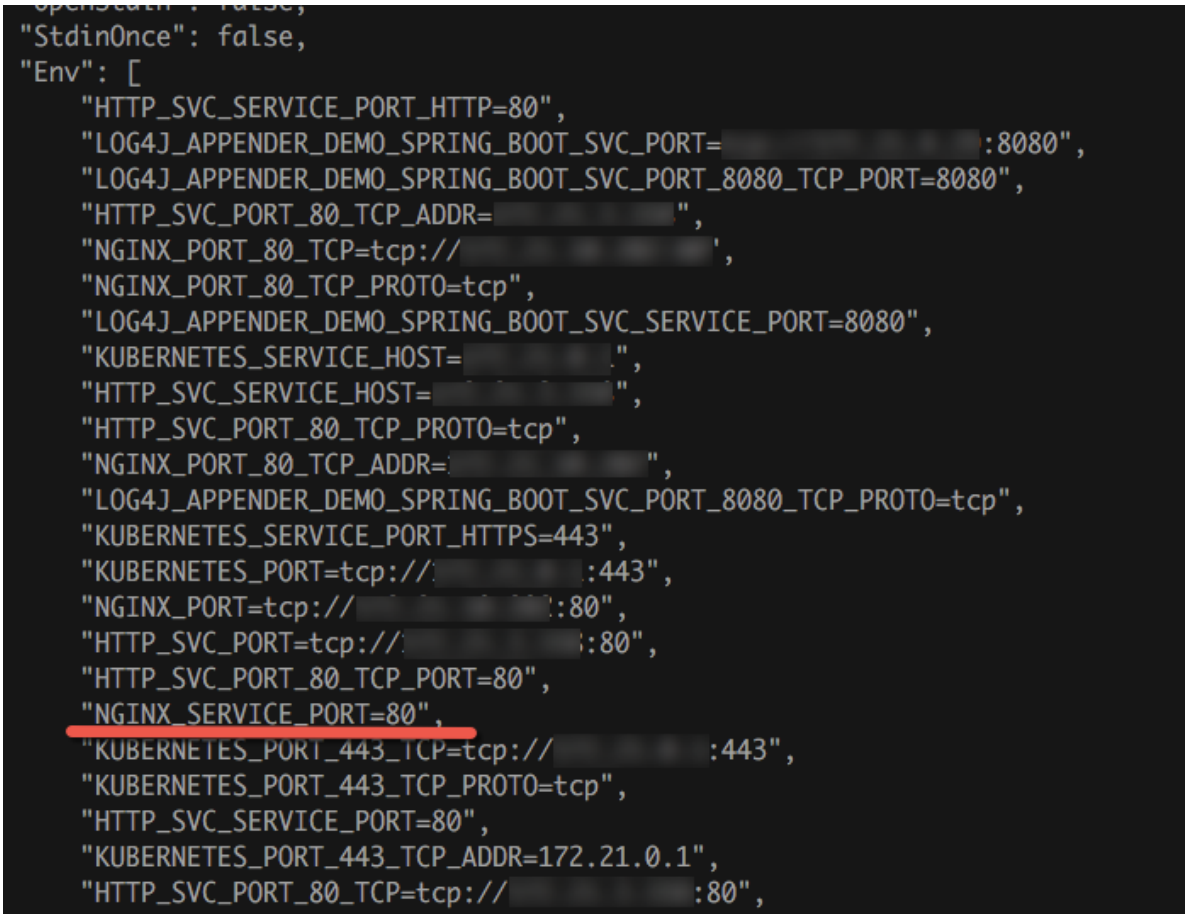
- **environment 配置方式**

  采集environment为NGINX_PORT_80_TCP_PORT=80且environment不为POD_NAMESPACE=kube-system的stdout以及stderr日志：

说明：

此处的environment为容器启动中配置的environment信息。

图 **4-34: environment** 配置方式示例



采集配置：

```
{
    "inputs": [
        {
            "type": "service_docker_stdout",
            "detail": {
                "Stdout": true,
                "Stderr": true,
                "IncludeEnv": {
                    "NGINX_PORT_80_TCP_PORT": "80"
                },
                "ExcludeEnv": {
                    "POD_NAMESPACE": "kube-system"
                }
            }
        }
    ]
```

```
}
```

- **label 配置方式**

  采集label为`io.kubernetes.container.name=nginx`且label不为`type=pre`的stdout以及stderr日志：

  > 说明：
  > 此处的label为docker inspect中的label信息，并不是Kubernetes配置中的label。

  图 **4-35: label配置方式示例**




```
{
    "inputs": [
        {
            "type": "service_docker_stdout",
            "detail": {
                "Stdout": true,
                "Stderr": true,
                "IncludeLabel": {
                    "io.kubernetes.container.name": "nginx"
                },
                "ExcludeLabel": {
                    "type": "pre"
                }
            }
        }
    ]
}
```

多行日志采集配置

多行日志采集对于Java异常堆栈输出的采集尤为重要，这里介绍一种标准的Java标准输出日志的采集配置。

- 日志示例

```
2018-02-03 14:18:41.968  INFO [spring-cloud-monitor] [nio-8080-exec-
4] c.g.s.web.controller.DemoController : service start
2018-02-03 14:18:41.969 ERROR [spring-cloud-monitor] [nio-8080-
exec-4] c.g.s.web.controller.DemoController : java.lang.NullPointe
rException
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(
ApplicationFilterChain.java:193)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(
ApplicationFilterChain.java:166)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWr
apperValve.java:199)
at org.apache.catalina.core.StandardContextValve.invoke(StandardCo
ntextValve.java:96)
...
2018-02-03 14:18:41.968  INFO [spring-cloud-monitor] [nio-8080-exec-
4] c.g.s.web.controller.DemoController : service start done
```

- 采集配置

  采集label为`app=monitor`输入的日志，行首为日期类型（为提高匹配效率，这里只判断行首的10个字节）。

```
{
"inputs": [
  {
    "detail": {
      "BeginLineCheckLength": 10,
      "BeginLineRegex": "\\d+-\\d+-\\d+.*",
      "IncludeLabel": {
        "app": "monitor"
      }
    },
    "type": "service_docker_stdout"
  }
]
}
```

采集数据处理

Logtail对于采集到的Docker标准输出，支持通用数据处理方式。建议使用基于上一节的多行日志格式，使用正则表达式将日志解析成time、module、thread、class、info。

- 采集配置：

  采集label为`app=monitor`输入的日志，行首为日期类型（为提高匹配效率，这里只判断行首的10个字节）。

```
{
"inputs": [
  {
    "detail": {
      "BeginLineCheckLength": 10,
      "BeginLineRegex": "\\d+-\\d+-\\d+.*",
```

```
      "IncludeLabel": {
        "app": "monitor"
      }
    },
    "type": "service_docker_stdout"
  }
],
"processors": [
    {
        "type": "processor_regex",
        "detail": {
            "SourceKey": "content",
            "Regex": "(\\d+-\\d+-\\d+ \\d+:\\d+:\\d+\\.\\d+)\\s+(\\w
+)\\s+\\[([^]]+)]\\s+\\[([^]]+)]\\s+:\\s+([\\s\\S]*)",
            "Keys": [
                "time",
                "module",
                "thread",
                "class",
                "info"
            ],
            "NoKeyError": true,
            "NoMatchError": true,
            "KeepSource": false
        }
    }
]
}
```

- 样例输出

  对于日志`2018-02-03 14:18:41.968 INFO [spring-cloud-monitor] [nio-8080-exec-4] c.g.s.web.controller.DemoController : service start done`处理后的输出为：

```
__tag__:__hostname__:logtail-dfgef
_container_name_:monitor
_image_name_:registry.cn-hangzhou.aliyuncs.xxxxxxxxxxxxxx
_namespace_:default
_pod_name_:monitor-6f54bd5d74-rtzc7
_pod_uid_:7f012b72-04c7-11e8-84aa-00163f00c369
_source_:stdout
_time_:2018-02-02T14:18:41.979147844Z
time:2018-02-02 02:18:41.968
level:INFO
module:spring-cloud-monitor
thread:nio-8080-exec-4
class:c.g.s.web.controller.DemoController
```

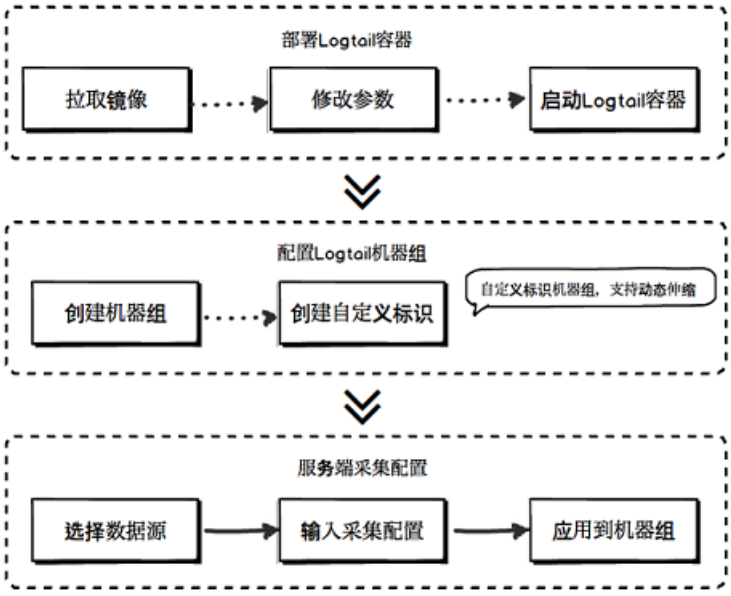
```
message:service start done
```

4.7.3 标准Docker日志采集流程

Logtail支持采集标准Docker日志，并附加容器的相关元数据信息一起上传到日志服务。

配置流程

图 **4-36:** 配置流程



1. 部署Logtail容器。
2. 配置Logtail机器组。

   日志服务控制台创建自定义标识机器组，后续该容器集群伸缩无需额外运维。
3. 创建服务端采集配置。

   在日志服务控制台创建采集配置，所有采集均为服务端配置，无需本地配置。

步骤一 部署Logtail容器

1. 拉取Logtail镜像。

   ```
   docker pull registry.cn-hangzhou.aliyuncs.com/log-service/logtail
   ```
2. 启动Logtail容器。

替换启动模板中的3个参数：${your_region_name}、${your_aliyun_user_id}和${your_machine_group_user_defined_id}。

```
docker run-d -v /:/logtail_host:ro -v /var/run/docker.sock:/var/run/docker.sock --env ALIYUN_LOGTAIL_CONFIG=/etc/ilogtail/conf/${your_region_name}/ilogtail_config.json --env ALIYUN_LOGTAIL_USER_ID=${your_aliyun_user_id} --env ALIYUN_LOGTAIL_USER_DEFINED_ID=${your_machine_group_user_defined_id} registry.cn-hangzhou.aliyuncs.com/log-service/logtail
```

 说明：

请在配置参数前执行以下任意一种配置，否则删除其他container时可能出现错误container text file busy。

- Centos 7.4 及以上版本设置fs.may_detach_mounts = 1，相关说明请参见*Bug 1468249*、*Bug 1441737*和*issue 34538*。
- 为Logtail授予privileged权限，启动参数中添加--privileged。详细内容请参考*docker run*命令。

参数	参数说明
${your_region_name}	日志服务Project所在Region，请根据网络类型输入正确的格式。包括： • 公网：*region*-internet。例如，华东一地域为cn-hangzhou-internet。 • 阿里云内网：*region*。例如，华东一地域为cn-hangzhou。 其中，*region*为*表 1*，请根据Project地域选择正确的参数。
${your_aliyun_user_id}	用户标识，请替换为您的阿里云主账号用户ID。主账号用户ID为字符串形式，如何查看ID请参考为非本账号*ECS*、自建*IDC*配置*AliUid*中的2.1节。
${your_machine_group_user_defined_id}	您集群的机器组自定义标识。如您尚未开启自定义标识，请参考创建用户自定义标识机器组的步骤一，开启userdefined-id。

```
docker run -d -v /:/logtail_host:ro -v /var/run/docker.sock:/var/run/docker.sock
```

```
--env ALIYUN_LOGTAIL_CONFIG=/etc/ilogtail/conf/cn_hangzhou/
ilogtail_config.json --env
 ALIYUN_LOGTAIL_USER_ID=1654218******--env ALIYUN_LOGTAIL_USER_
DEFINED_ID=log-docker-demo registry.cn-hangzhou.aliyuncs.com/log-
service/logtail
```

 说明：

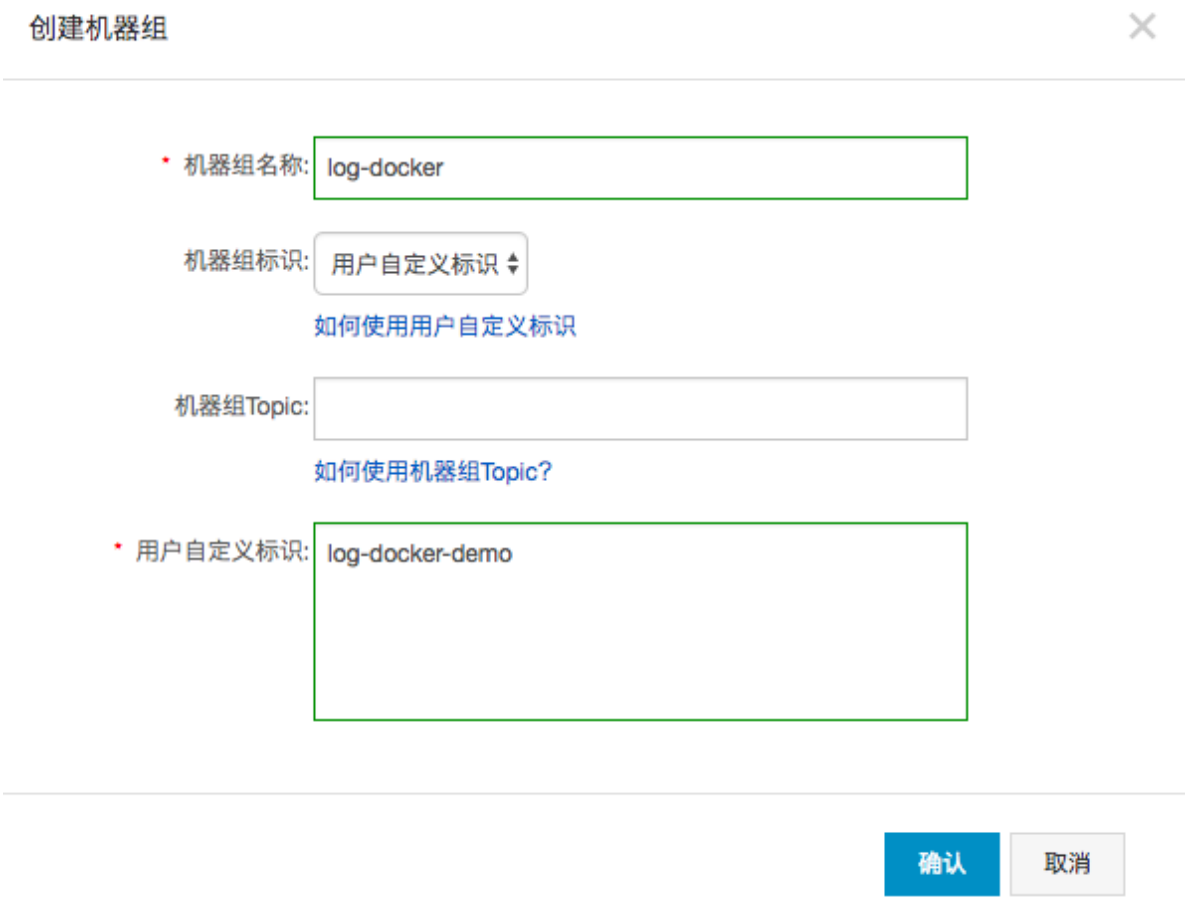
您可以自定义配置Logtail容器的启动参数，只需保证以下前提条件：

1. 启动时，必须具备3个环境变量：`ALIYUN_LOGTAIL_USER_DEFINED_ID`、`ALIYUN_LOGTAIL_USER_ID`、`ALIYUN_LOGTAIL_CONFIG`。
2. 必须将Docker的Domain Socket挂载到`/var/run/docker.sock`。
3. 如果您需要采集容器标准输出、容器或宿主机文件，需要将根目录挂载到Logtail容器的`/logtail_host`目录。
4. 如果Logtail日志*`/usr/local/ilogtail/ilogtail.LOG`*中出现`The parameter is invalid : uuid=none`的错误日志，请在宿主机上创建一个*`product_uuid`*文件，在其中输入任意合法UUID（例如`169E98C9-ABC0-4A92-B1D2-AA6239C0D261`），并把该文件挂载到Logtail容器的*`/sys/class/dmi/id/product_uuid`*路径上。

步骤2 配置机器组

1. 开通日志服务并创建Project、Logstore，详细步骤请参考准备流程。
2. 在日志服务控制台的机器组列表页面单击创建*IP*地址机器组。
3. 选择用户自定义标识，将您上一步配置的 `ALIYUN_LOGTAIL_USER_DEFINED_ID`填入用户自定义标识内容框中。

图 4-37: 配置机器组



配置完成一分钟后，在机器组列表页面单击右侧的查看状态按钮，即可看到已经部署Logtail容器的心跳状态。具体参见管理机器组中的查看状态部分。

步骤3 创建采集配置

请根据您的需求在控制台创建Logtail采集配置，采集配置步骤请参考：

- 容器内文本文件#推荐#
- 容器标准输出#推荐#
- 宿主机文本文件

  默认宿主机根目录挂载到Logtail容器的`/logtail_host`目录，配置路径时，您需要加上此前缀。例如需要采集宿主机上`/home/logs/app_log/`目录下的数据，配置页面中日志路径设置为`/logtail_host/home/logs/app_log/`。
- *Syslog*
- 简介

其他操作

- 查看Logtail容器运行状态

  您可以执行命令`docker exec ${logtail_container_id} /etc/init.d/ilogtaild status`查看Logtail运行状态。

- 查看Logtail的版本号信息、IP、启动时间等

  您可以执行命令`docker exec ${logtail_container_id} cat /usr/local/ilogtail/app_info.json`查看Logtail相关信息。

- 查看Logtail的运行日志

  Logtail运行日志保存在`/usr/local/ilogtail/`目录下，文件名为`ilogtail.LOG`，轮转文件会压缩存储为`ilogtail.LOG.x.gz`。

  示例如下：

```
[root@iZbp17enxc2us3624wexh2Z ilogtail]# docker exec a287de895e40
tail -n 5 /usr/local/ilogtail/ilogtail.LOG
[2018-02-06 08:13:35.721864]    [INFO]    [8]    [build/release64/
sls/ilogtail/LogtailPlugin.cpp:104]    logtail plugin Resume:start
[2018-02-06 08:13:35.722135]    [INFO]    [8]    [build/release64/
sls/ilogtail/LogtailPlugin.cpp:106]    logtail plugin Resume:success
[2018-02-06 08:13:35.722149]    [INFO]    [8]    [build/release64
/sls/ilogtail/EventDispatcher.cpp:369]    start add existed check
point events, size:0
[2018-02-06 08:13:35.722155]    [INFO]    [8]    [build/release64
/sls/ilogtail/EventDispatcher.cpp:511]    add existed check point
events, size:0    cache size:0    event size:0    success count:0
[2018-02-06 08:13:39.725417]    [INFO]    [8]    [build/release64/
sls/ilogtail/ConfigManager.cpp:3776]    check container path update
flag:0    size:1
```

  容器stdout并不具备参考意义，请忽略以下stdout输出：

```


start umount useless mount points, /shm$|/merged$|/mqueue$
umount: /logtail_host/var/lib/docker/overlay2/3fd0043af174cb0273c3
c7869500fbe2bdb95d13b1e110172ef57fe840c82155/merged: must be
superuser to unmount
umount: /logtail_host/var/lib/docker/overlay2/d5b10aa19399992755de
1f85d25009528daa749c1bf8c16edff44beab6e69718/merged: must be
superuser to unmount
umount: /logtail_host/var/lib/docker/overlay2/5c3125daddacedec29df
72ad0c52fac800cd56c6e880dc4e8a640b1e16c22dbe/merged: must be
superuser to unmount
......
xargs: umount: exited with status 255; aborting
umount done
start logtail
ilogtail is running
logtail status:
```

```
ilogtail is running
```

- 重启Logtail

  请参考以下示例重启Logtail。

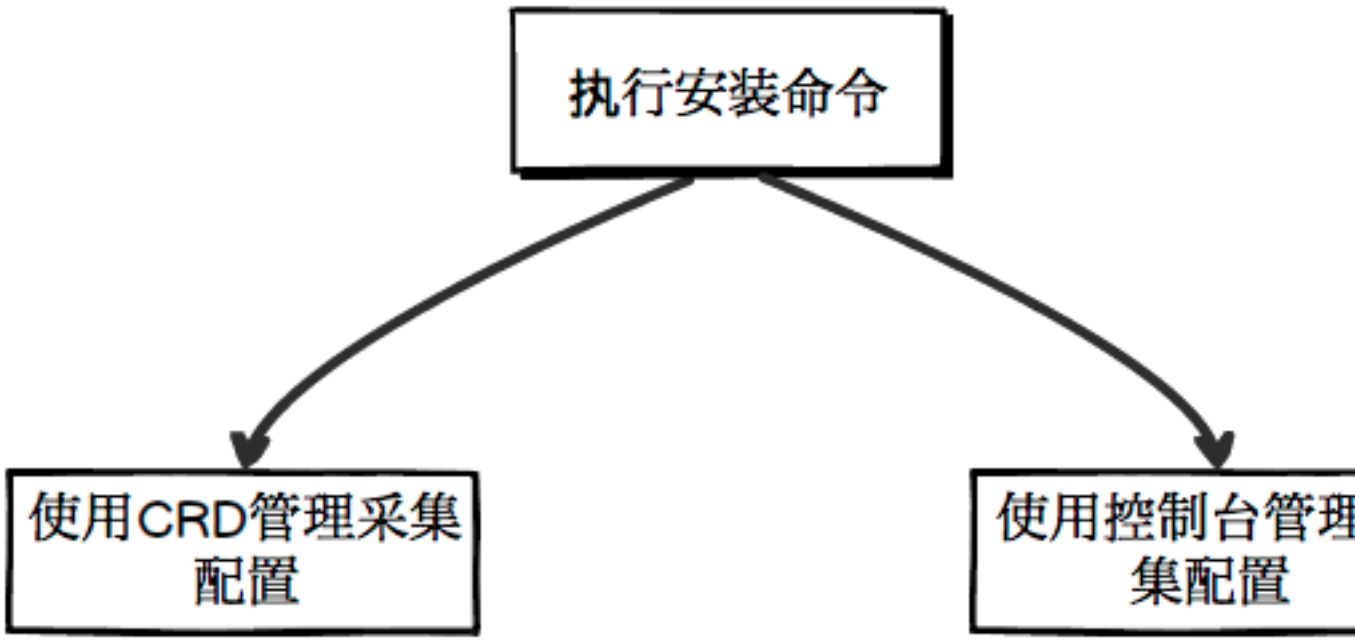
```
[root@iZbp17enxc2us3624wexh2Z ilogtail]# docker exec a287de895e40 /
etc/init.d/ilogtaild stop
kill process Name: ilogtail pid: 7
kill process Name: ilogtail pid: 8
stop success
[root@iZbp17enxc2us3624wexh2Z ilogtail]# docker exec a287de895e40 /
etc/init.d/ilogtaild start
ilogtail is running
```

4.7.4 Kubernetes日志采集流程

日志服务支持通过Logtail采集Kubernetes集群日志，并支持CRD（CustomResourceDefinition）进行采集配置管理。本文主要介绍如何安装并使用Logtail采集Kubernetes集群日志。

配置流程

图 **4-38:** 配置流程



1. 执行安装命令，安装alibaba-log-controller Helm包。
2. 根据您的需求选择使用CRD（CustomResourceDefinition）或控制台进行采集配置管理。

视频教程

https://cloud.video.taobao.com/play/u/3220778205/p/1/e/6/t/1/50076466637.mp4

步骤1 安装Logtail

阿里云容器服务Kubernetes安装方式

安装步骤

1. 登录您的阿里云容器服务Kubernetes的Master节点。登录方式请参考*SSH访问Kubernetes集群*。
2. 将下述命令中的`${your_k8s_cluster_id}`替换为您的Kubernetes集群id，并执行此命令。

```
wget http://logtail-release-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com
/kubernetes/alicloud-log-k8s-install.sh -O alicloud-log-k8s-install.
sh; chmod 744 ./alicloud-log-k8s-install.sh; sh ./alicloud-log-k8s-
install.sh ${your_k8s_cluster_id}
```

安装后，日志服务会自动创建和您的Kubernetes集群处于同一region的日志服务Project，Project名称为`k8s-log-${your_k8s_cluster_id}`；同时会在该Project下创建机器组，机器组名为`k8s-group-${your_k8s_cluster_id}`。



说明：

- Project `k8s-log-${your_k8s_cluster_id}`下会自动创建名为`config-operation-log`的Logstore，用于存储alibaba-log-controller的运行日志。请勿删除此Logstore，否则无法为alibaba-log-controller排查问题。
- 若您需要将日志采集到已有的Project，请执行安装命令`sh ./alicloud-log-k8s-install.sh *${your_k8s_cluster_id} ${your_project_name}*` ，并确保日志服务Project和您的Kubernetes集群在同一地域。

安装示例

示例如下，执行成功后将会输出以下内容：

```
[root@iZbp******biaZ ~]# wget http://logtail-release-cn-hangzhou.oss
-cn-hangzhou.aliyuncs.com/kubernetes/alicloud-log-k8s-install.sh -O
alicloud-log-k8s-install.sh; chmod 744 ./alicloud-log-k8s-install.sh;
sh ./alicloud-log-k8s-install.sh c12ba20**************86939f0b
....
....
....
alibaba-cloud-log/Chart.yaml
alibaba-cloud-log/templates/
alibaba-cloud-log/templates/_helpers.tpl
alibaba-cloud-log/templates/alicloud-log-crd.yaml
```

```
alibaba-cloud-log/templates/logtail-daemonset.yaml
alibaba-cloud-log/templates/NOTES.txt
alibaba-cloud-log/values.yaml
NAME:   alibaba-log-controller
LAST DEPLOYED: Wed May 16 18:43:06 2018
NAMESPACE: default
STATUS: DEPLOYED
RESOURCES:
==> v1beta1/ClusterRoleBinding
NAME                    AGE
alibaba-log-controller  0s
==> v1beta1/DaemonSet
NAME     DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR
  AGE
logtail  2        2        0      2           0          <none>
  0s
==> v1beta1/Deployment
NAME                    DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
alibaba-log-controller  1        1        1           0          0s
==> v1/Pod(related)
NAME                                    READY  STATUS
RESTARTS  AGE
logtail-ff6rf                           0/1    ContainerCreating  0
       0s
logtail-q5s87                           0/1    ContainerCreating  0
       0s
alibaba-log-controller-7cf6d7dbb5-qvn6w  0/1    ContainerCreating  0
       0s
==> v1/ServiceAccount
NAME                    SECRETS  AGE
alibaba-log-controller  1        0s
==> v1beta1/CustomResourceDefinition
NAME                                AGE
aliyunlogconfigs.log.alibabacloud.com  0s
==> v1beta1/ClusterRole
alibaba-log-controller  0s
[SUCCESS] install helm package : alibaba-log-controller success.
```

您可以使用`helm status alibaba-log-controller`查看Pod当前状态，若状态全部成功后，表示安装成功。

安装成功后登录日志服务控制台，即可看到已经自动创建出的日志服务Project（若您的Project数过多，可以搜索`k8s-log`关键字）。

自建Kubernetes安装方式

前提条件

1. Kubernetes集群版本1.8及以上。
2. 已经安装Helm命令，版本2.6.4及以上。

安装步骤

1. 在日志服务控制台创建一个Project，Project名称以`k8s-log-custom-`开头。

2. 将下述命令中的参数替换，并执行此命令。

```
wget http://logtail-release-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com
/kubernetes/alicloud-log-k8s-custom-install.sh; chmod 744 ./alicloud
-log-k8s-custom-install.sh; sh ./alicloud-log-k8s-custom-install.sh
 {your-project-suffix} {region-id} {aliuid} {access-key-id} {access-
key-secret}
```

各参数及其说明如下：

参数	说明
{your-project-suffix}	您在第二步创建的Project名称的`k8s-log-custom-`之后部分。例如创建的Project为`k8s-log-custom-xxxx`，这边填写`xxxx`。
{regionId}	您的Project所在区域的Region Id，请在服务入口中查找，例如华东 1（杭州）的Region Id为`cn-hangzhou`。
{aliuid}	用户标识（AliUid），请替换为您的阿里云主账号用户ID。主账号用户ID为字符串形式，如何查看ID请参考用户标识配置中的2.1节。
{access-key-id}	您的账号access key id。推荐使用子账号access key，并授予AliyunLogFullAccess权限，具体设置参考简介。
{access-key-secret}	您的账号access key secret。推荐使用子账号access key，并授予AliyunLogFullAccess权限，具体设置参考简介。

安装好之后，日志服务会自动在该Project下创建机器组，机器组名为`k8s-group-${your_k8s_cluster_id}`。



说明：

- Project下会自动创建名为`config-operation-log`的Logstore，请不要删除此Logstore。
- 自建Kubernetes安装时，默认为Logtail授予`privileged`权限，主要为避免删除其他POD时可能出现错误`container text file busy`。相关说明请参考：*Bug 1468249*、*Bug 1441737*和 *issue 34538*。

安装示例

示例如下，执行成功后将会输出以下内容：

```
[root@iZbp1dsxxxxxqfbiaZ ~]#  wget http://logtail-release-cn-hangzhou
.oss-cn-hangzhou.aliyuncs.com/kubernetes/alicloud-log-k8s-custom-
install.sh; chmod 744 ./alicloud-log-k8s-custom-install.sh; sh ./
```

```
alicloud-log-k8s-custom-install.sh xxxx cn-hangzhou 165xxxxxxx050
LTAxxxxxxxxxx AIxxxxxxxxxxxxxxxxxxxxxxxxxxxxxe
....
....
....
NAME:   alibaba-log-controller
LAST DEPLOYED: Fri May 18 16:52:38 2018
NAMESPACE: default
STATUS: DEPLOYED
RESOURCES:
==> v1beta1/ClusterRoleBinding
NAME                    AGE
alibaba-log-controller  0s
==> v1beta1/DaemonSet
NAME        DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE
SELECTOR  AGE
logtail-ds  2        2        0      2           0          <none>
     0s
==> v1beta1/Deployment
NAME                    DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
alibaba-log-controller  1        1        1           0          0s
==> v1/Pod(related)
NAME                                    READY  STATUS
RESTARTS  AGE
logtail-ds-7xf2d                        0/1    ContainerCreating  0
       0s
logtail-ds-9j4bx                        0/1    ContainerCreating  0
       0s
alibaba-log-controller-796f8496b6-6jxb2  0/1    ContainerCreating  0
       0s
==> v1/ServiceAccount
NAME                    SECRETS  AGE
alibaba-log-controller  1        0s
==> v1beta1/CustomResourceDefinition
NAME                                AGE
aliyunlogconfigs.log.alibabacloud.com  0s
==> v1beta1/ClusterRole
alibaba-log-controller  0s
[INFO] your k8s is using project : k8s-log-custom-xxx, region : cn-
hangzhou, aliuid : 1654218965343050, accessKeyId : LTAxxxxxxxxxx
[SUCCESS] install helm package : alibaba-log-controller success.
```

您可以使用`helm status alibaba-log-controller`查看Pod当前状态，若状态全部成功后，表示安装成功。

安装成功后登录日志服务控制台，即可看到已经自动创建出的日志服务Project（若您的Project数过多，可以搜索`k8s-log`关键字）。

步骤二 配置

日志采集配置默认支持控制台配置方式，同时针对Kubernetes微服务开发模式，我们还提供CRD的配置方式，您可以直接使用kubectl对配置进行管理。以下是对两种配置方式进行的比较：

-	CRD方式	控制台方式
操作复杂度	低	一般

-	CRD方式	控制台方式
功能项	支持除控制台方式外的高级配置	一般
上手难度	一般	低
网络连接	连接Kubernetes集群	连接互联网
与组件部署集成	支持	不支持
鉴权方式	Kubernetes鉴权	云账号鉴权

推荐您使用CRD方式进行采集配置管理，该方式与Kubernetes部署、发布流程的集成更加完善。

通过控制台管理采集配置

请根据您的需求在控制台创建Logtail采集配置，采集配置步骤请参考：

- 容器内文本文件#推荐#
- 容器标准输出#推荐#
- 宿主机文本文件

  默认宿主机根目录挂载到Logtail容器的`/logtail_host`目录，配置路径时，您需要加上此前缀。例如需要采集宿主机上`/home/logs/app_log/`目录下的数据，配置页面中日志路径设置为`/logtail_host/home/logs/app_log/`。
- *Syslog*
- 简介

通过CRD管理采集配置

针对Kubernetes微服务开发模式，日志服务同时提供CRD的配置方式，您可以直接使用kubectl对配置进行管理，该方式与Kubernetes部署、发布流程的集成更加完善。

详细说明请参考*Kubernetes-CRD*配置日志采集。

其他操作

DaemonSet部署方式迁移步骤

如果您之前使用的DaemonSet方式部署的日志服务Logtail，将无法使用CRD的方式进行配置管理。您可以通过以下方式迁移到新的版本：



说明：

升级期间会有部分日志重复；CRD配置管理方式只对使用CRD创建的配置生效（由于历史配置使用非CRD方式创建，因此历史配置不支持CRD管理方式）。

1. 按照新版本的方式安装，安装命令最后新增一个参数为您之前Kubernetes集群使用的日志服务Project名。

   例如Project名为`k8s-log-demo`，集群id为`c12ba2028cxxxxxxxxxx6939f0b`，安装命令为：

   ```
   wget http://logtail-release-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com
   /kubernetes/alicloud-log-k8s-install.sh -O alicloud-log-k8s-install.
   sh; chmod 744 ./alicloud-log-k8s-install.sh; sh ./alicloud-log-k8s-
   install.sh c12ba2028cxxxxxxxxxx6939f0b k8s-log-demo
   ```

2. 安装成功后，进入日志服务控制台，将历史采集配置应用到新的机器组`k8s-group-${your_k8s_cluster_id}`。
3. 一分后，将历史采集配置从历史的机器组中解绑。
4. 观察日志采集正常后，可以选择删除之前安装的logtail daemonset。

多集群使用同一个日志服务Project

如果您希望多个集群将日志采集到同一个日志服务Project，您可以在安装其他集群日志服务组件时，将上述安装参数中的`${your_k8s_cluster_id}`替换为您第一次安装的集群ID。

例如您现在有3个集群，ID分别为 abc001、abc002、abc003，三个集群安装组件的参数中`${your_k8s_cluster_id}`都填写为`abc001`。



说明：

此方式不支持跨region的Kubernetes多集群共享。

Logtail容器日志

Logtail日志存储在Logtail容器中的`/usr/local/ilogtail/`目录中，文件名为`ilogtail.LOG`以及`logtail_plugin.LOG`，容器stdout并不具备参考意义，请忽略以下stdout输出：

```
start umount useless mount points, /shm$|/merged$|/mqueue$
umount: /logtail_host/var/lib/docker/overlay2/3fd0043af174cb0273c3
c7869500fbe2bdb95d13b1e110172ef57fe840c82155/merged: must be superuser
 to unmount
umount: /logtail_host/var/lib/docker/overlay2/d5b10aa19399992755de
1f85d25009528daa749c1bf8c16edff44beab6e69718/merged: must be superuser
 to unmount
umount: /logtail_host/var/lib/docker/overlay2/5c3125daddacedec29df
72ad0c52fac800cd56c6e880dc4e8a640b1e16c22dbe/merged: must be superuser
 to unmount
......
```

```
xargs: umount: exited with status 255; aborting
umount done
start logtail
ilogtail is running
logtail status:
ilogtail is running
```

查看Kubernetes集群中日志相关组件状态

```
helm status alibaba-log-controller
```

alibaba-log-controller启动失败

请确认您是按照以下方式执行的安装：

1. 安装命令在Kubernetes集群的master节点执行。
2. 安装命令参数输入的是您的集群ID。

若由于以上问题安装失败，请使用`helm del --purge alibaba-log-controller`删除安装包并重新执行安装。

若以上操作依然安装失败，请提交工单至日志服务。

查看Kubernetes集群中Logtail DaemonSet状态。

您可以执行命令`kubectl get ds -n kube-system`查看Logtail运行状态。



说明：

Logtail 默认的namespace为`kube-system`。

查看Logtail的版本号信息、IP、启动时间

示例如下：

```
[root@iZbp1dsu6v77zfb40qfbiaZ ~]# kubectl get po -n kube-system | grep
 logtail
NAME            READY     STATUS    RESTARTS   AGE
logtail-ds-gb92k   1/1       Running   0          2h
logtail-ds-wm7lw   1/1       Running   0          4d
[root@iZbp1dsu6v77zfb40qfbiaZ ~]# kubectl exec logtail-ds-gb92k -n
kube-system cat /usr/local/ilogtail/app_info.json
{
   "UUID" : "",
   "hostname" : "logtail-ds-gb92k",
   "instance_id" : "0EBB2B0E-0A3B-11E8-B0CE-0A58AC140402_172.20.4.
2_1517810940",
   "ip" : "172.20.4.2",
   "logtail_version" : "0.16.2",
   "os" : "Linux; 3.10.0-693.2.2.el7.x86_64; #1 SMP Tue Sep 12 22:26:
13 UTC 2017; x86_64",
   "update_time" : "2018-02-05 06:09:01"
```

```
}
```

查看Logtail的运行日志

Logtail运行日志保存在/usr/local/ilogtail/目录下，文件名为ilogtail.LOG，轮转文件会压缩存储为ilogtail.LOG.x.gz。

示例如下：

```
[root@iZbp1dsu6v77zfb40qfbiaZ ~]# kubectl exec logtail-ds-gb92k -n
kube-system tail /usr/local/ilogtail/ilogtail.LOG
[2018-02-05 06:09:02.168693] [INFO] [9] [build/release64/sls/ilogtail/
LogtailPlugin.cpp:104] logtail plugin Resume:start
[2018-02-05 06:09:02.168807] [INFO] [9] [build/release64/sls/ilogtail/
LogtailPlugin.cpp:106] logtail plugin Resume:success
[2018-02-05 06:09:02.168822] [INFO] [9] [build/release64/sls/ilogtail/
EventDispatcher.cpp:369] start add existed check point events, size:0
[2018-02-05 06:09:02.168827] [INFO] [9] [build/release64/sls/ilogtail/
EventDispatcher.cpp:511] add existed check point events, size:0 cache
size:0 event size:0 success count:0
```

重启某个pod的Logtail

示例如下：

```
[root@iZbp1dsu6v77zfb40qfbiaZ ~]# kubectl exec logtail-ds-gb92k -n
kube-system /etc/init.d/ilogtaild stop
kill process Name: ilogtail pid: 7
kill process Name: ilogtail pid: 9
stop success
[root@iZbp1dsu6v77zfb40qfbiaZ ~]# kubectl exec logtail-ds-gb92k -n
kube-system /etc/init.d/ilogtaild start
ilogtail is running
```

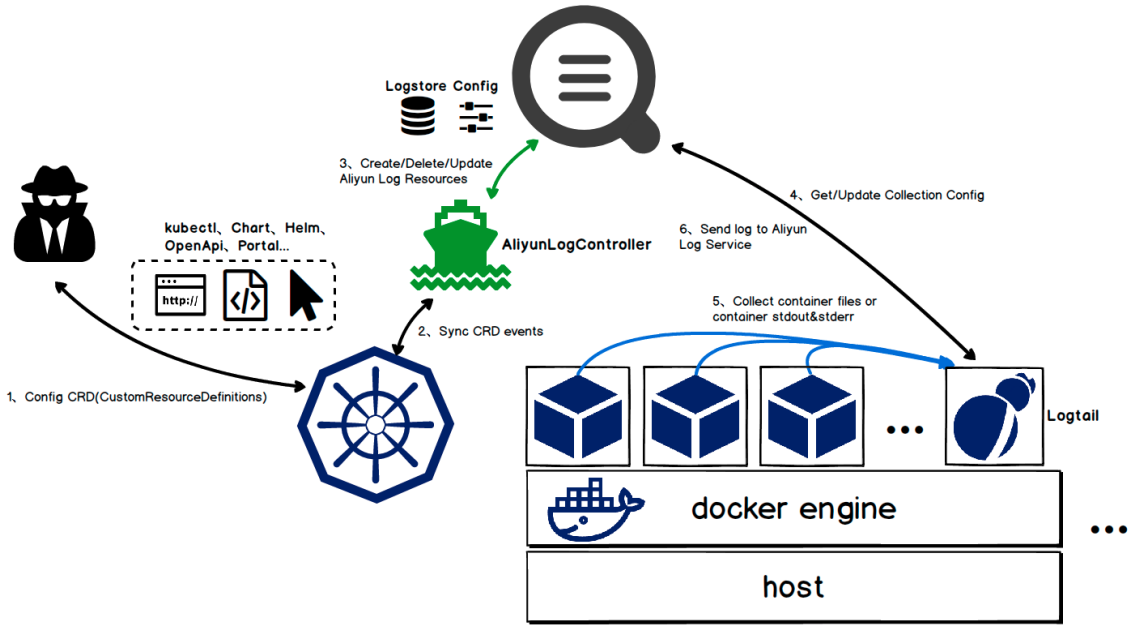
4.7.5 Kubernetes-CRD配置日志采集

配置日志采集默认通过控制台方式配置。针对Kubernetes微服务开发模式，除控制台方式外，日志服务还提供CRD的配置方式日志，您可以直接使用kubectl对配置进行管理。

推荐您使用CRD方式进行采集配置管理，该方式与Kubernetes部署、发布流程的集成更加完善。

实现原理

图 **4-39:** 实现原理



执行安装命令时会自动安装`alibaba-log-controller`的Helm包。Helm包中主要执行了以下操作：

1. 创建aliyunlogconfigs CRD（CustomResourceDefinition）。
2. 部署alibaba-log-controller的Deployment。
3. 部署Logtail的DaemonSet。

具体配置的内部工作流程如下：

1. 用户使用`kubectl`或其他工具应用aliyunlogconfigs CRD配置。
2. alibaba-log-controller监听到配置更新。
3. alibaba-log-controller根据CRD内容以及服务端状态，自动向日志服务提交logstore创建、配置创建以及应用机器组的请求。
4. 以DaemonSet模式运行的Logtail会定期请求配置服务器，获取新的或已更新的配置并进行热加载。
5. Logtail根据配置信息采集各个容器(POD)上的标准输出或文件。
6. 最终Logtail将处理、聚合好的数据发送到日志服务。

配置方式

> 说明：
> 如果您之前使用的DaemonSet方式部署的日志服务Logtail，将无法使用CRD的方式进行配置管理。升级方式参见本文档中**DaemonSet部署方式迁移步骤**。

您只需要定义AliyunLogConfig的CRD即可实现配置的创建，删除对应的CRD资源即可删除对应配置。CRD配置格式如下：

```
apiVersion: log.alibabacloud.com/v1alpha1       ## 默认值，无需改变
kind: AliyunLogConfig                           ## 默认值，无需改变
metadata:
  name: simple-stdout-example                   ## 资源名，必须在集群内唯一
spec:
  logstore: k8s-stdout                          ## Logstore名，不存在时自动创建
  shardCount: 2                                 ## [可选] logstore分区数，默认为2，支持1-10
  lifeCycle: 90                                 ## [可选] 该logstore存储时间，默认为90，支持1-7300，7300天为永久存储
  logtailConfig:                                ## 详细配置
    inputType: plugin                           ## 采集的输入类型，一般为file或plugin
    configName: simple-stdout-example           ## 采集配置名，需要和资源名(metadata.name)一致
    inputDetail:                                ## 详细配置信息，具体请参考示例
      ...
```

配置完成并应用配置后，会自动创建alibaba-log-controller。

查看配置

您可以通过Kubernetes CRD或控制台查看配置。

控制台查看配置参见管理采集配置。

> 说明：
> 使用CRD管理方式，若您在控制台更改配置，下一次执行CRD更新配置时，会覆盖控制台的更改内容。

- 使用`kubectl get aliyunlogconfigs`查看当前所有配置。

```
[root@iZbp1dsbiaZ ~]# kubectl get aliyunlogconfigs
NAME                   AGE
regex-file-example     10s
regex-stdout-example   4h
```

```
simple-file-example    5s
```

- 使用`kubectl get aliyunlogconfigs ${config_name} -o yaml`查看详细配置和状态。

  配置中`status`字段显示配置执行的结果，若配置应用成功，`status`字段中的`statusCode`为200。若`statusCode`非200，则说明配置应用失败。

```
[root@iZbp1dsbiaZ ~]# kubectl get aliyunlogconfigs simple-file-
example -o yaml
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
annotations:
  kubectl.kubernetes.io/last-applied-configuration: |
    {"apiVersion":"log.alibabacloud.com/v1alpha1","kind":"AliyunLogC
onfig","metadata":{"annotations":{},"name":"simple-file-example","
namespace":"default"},"spec":{"logstore":"k8s-file","logtailConfig
":{"configName":"simple-file-example","inputDetail":{"dockerFile
":true,"dockerIncludeEnv":{"ALIYUN_LOGTAIL_USER_DEFINED_ID":""},"
filePattern":"simple.LOG","logPath":"/usr/local/ilogtail","logType
":"common_reg_log"},"inputType":"file"}}}
clusterName: ""
creationTimestamp: 2018-05-17T08:44:46Z
generation: 0
name: simple-file-example
namespace: default
resourceVersion: "21790443"
selfLink: /apis/log.alibabacloud.com/v1alpha1/namespaces/default/
aliyunlogconfigs/simple-file-example
uid: 8d3a09c4-59ae-11e8-851d-00163f008685
spec:
lifeCycle: null
logstore: k8s-file
logtailConfig:
  configName: simple-file-example
  inputDetail:
    dockerFile: true
    dockerIncludeEnv:
      ALIYUN_LOGTAIL_USER_DEFINED_ID: ""
    filePattern: simple.LOG
    logPath: /usr/local/ilogtail
    logType: common_reg_log
  inputType: file
machineGroups: null
project: ""
shardCount: null
status:
status: OK
```

```
statusCode: 200
```

配置示例

容器标准输出

容器标准输出中，需要将`inputType`设置为`plugin`，并将具体信息填写到`inputDetail`下的`plugin`字段，详细配置字段及其含义请参考容器标准输出。

- 极简采集方式

  采集除了环境变量中配置`COLLECT_STDOUT_FLAG=false`之外所有容器的标准输出(stdout和stderr)。

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # your config name, must be unique in you k8s cluster
  name: simple-stdout-example
spec:
  # logstore name to upload log
  logstore: k8s-stdout
  # logtail config detail
  logtailConfig:
    # docker stdout's input type is 'plugin'
    inputType: plugin
    # logtail config name, should be same with [metadata.name]
    configName: simple-stdout-example
    inputDetail:
      plugin:
        inputs:
          -
            # input type
            type: service_docker_stdout
            detail:
              # collect stdout and stderr
              Stdout: true
              Stderr: true
              # collect all container's stdout except containers
with "COLLECT_STDOUT_FLAG:false" in docker env config
              ExcludeEnv:
                COLLECT_STDOUT_FLAG: "false"
```

- 自定义处理采集方式

  采集grafana的access log，并将access log解析成结构化数据。

  grafana的容器配置中包含环境变量为：`GF_INSTALL_PLUGINS=grafana-piechart-....`，通过配置`IncludeEnv`为`GF_INSTALL_PLUGINS: ''`指定Logtail只采集该容器的标准输出。

图 **4-40:** 自定义处理采集方式



grafana的access log格式如下：

```
t=2018-03-09T07:14:03+0000 lvl=info msg="Request Completed" logger
=context userId=0 orgId=0 uname= method=GET path=/ status=302
remote_addr=172.16.64.154 time_ms=0 size=29 referer=
```

使用正则表达式解析access log，具体配置如下：

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # your config name, must be unique in you k8s cluster
  name: regex-stdout-example
spec:
  # logstore name to upload log
  logstore: k8s-stdout-regex
  # logtail config detail
  logtailConfig:
    # docker stdout's input type is 'plugin'
    inputType: plugin
    # logtail config name, should be same with [metadata.name]
    configName: regex-stdout-example
    inputDetail:
      plugin:
        inputs:
          -
            # input type
            type: service_docker_stdout
            detail:
              # 只采集stdout，不采集stderr
              Stdout: true
              Stderr: false
              # 只采集容器环境变量中配置key为"GF_INSTALL_PLUGINS"的stdout

              IncludeEnv:
                GF_INSTALL_PLUGINS: ''
        processors:
          -
            # 使用正则表达式处理
            type: processor_regex
            detail:
              # docker 采集的数据默认key为"content"
```

```
                SourceKey: content
                # 正则表达式提取
                Regex: 't=(\d+-\d+-\w+:\d+:\d+\+\d+) lvl=(\w+) msg
="([^"]+)" logger=(\w+) userId=(\w+) orgId=(\w+) uname=(\S*) method
=(\w+) path=(\S+) status=(\d+) remote_addr=(\S+) time_ms=(\d+) size
=(\d+) referer=(\S*).*'
                # 提取出的key
                Keys: ['time', 'level', 'message', 'logger', 'userId
', 'orgId', 'uname', 'method', 'path', 'status', 'remote_addr', '
time_ms', 'size', 'referer']
                # 保留原始字段
                KeepSource: true
                NoKeyError: true
                NoMatchError: true
```

配置应用后，采集到日志服务的数据如下：

图 **4-41:** 采集到的日志数据



容器文件

- 极简文件

采集环境变量配置中含有key为`ALIYUN_LOGTAIL_USER_DEFINED_ID`的容器内日志文件，文件所处的路径为`/data/logs/app_1`，文件名为`simple.LOG`。

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # your config name, must be unique in you k8s cluster
  name: simple-file-example
spec:
  # logstore name to upload log
  logstore: k8s-file
  # logtail config detail
  logtailConfig:
    # log file's input type is 'file'
    inputType: file
    # logtail config name, should be same with [metadata.name]
    configName: simple-file-example
    inputDetail:
      # 极简模式日志，logType设置为"common_reg_log"
      logType: common_reg_log
      # 日志文件夹
```

```
      logPath: /data/logs/app_1
      # 文件名, 支持通配符，例如log_*.log
      filePattern: simple.LOG
      # 采集容器内的文件，dockerFile flag设置为true
      dockerFile: true
      # only collect container with "ALIYUN_LOGTAIL_USER_DEFINED_ID
" in docker env config
      dockerIncludeEnv:
        ALIYUN_LOGTAIL_USER_DEFINED_ID: ""
```

- 完整正则模式文件

  对于某Java程序日志样例为：

```
[2018-05-11T20:10:16,000] [INFO] [SessionTracker] [SessionTra
ckerImpl.java:148] Expiring sessions
java.sql.SQLException: Incorrect string value: '\xF0\x9F\x8E\x8F
",...' for column 'data' at row 1
at org.springframework.jdbc.support.AbstractFallbackSQLExceptionTr
anslator.translate(AbstractFallbackSQLExceptionTranslator.java:84)
at org.springframework.jdbc.support.AbstractFallbackSQLException
```

  日志中由于包含错误堆栈信息，可能一条日志会被分解成多行，因此需要设置行首正则表达式；为了提取出各个字段，这里我们使用正则表达式进行提取，具体配置如下：

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # your config name, must be unique in you k8s cluster
  name: regex-file-example
spec:
  # logstore name to upload log
  logstore: k8s-file
  logtailConfig:
    # log file's input type is 'file'
    inputType: file
    # logtail config name, should be same with [metadata.name]
    configName: regex-file-example
    inputDetail:
      # 对于正则类型的日志，将logType设置为common_reg_log
      logType: common_reg_log
      # 日志文件夹
      logPath: /app/logs
      # 文件名, 支持通配符，例如log_*.log
      filePattern: error.LOG
      # 行首正则表达式
      logBeginRegex: '\[\d+-\d+-\w+:\d+:\d+,\d+]\s\[\w+]\s.*'
      # 解析正则
      regex: '\[([^]]+)]\s\[(\w+)]\s\[(\w+)]\s\[([^:]+):(\d+)]\s
(.*)'
      # 提取出的key列表
      key : ["time", "level", "method", "file", "line", "message"]
      # 正则模式日志，时间解析默认从日志中的`time`提取，如果无需提取时间，可不
设置该字段
      timeFormat: '%Y-%m-%dT%H:%M:%S'
      # 采集容器内的文件，dockerFile flag设置为true
      dockerFile: true
```

```
      # only collect container with "ALIYUN_LOGTAIL_USER_DEFINED_ID
" in docker env config
      dockerIncludeEnv:
        ALIYUN_LOGTAIL_USER_DEFINED_ID: ""
```

配置应用后，采集到日志服务的数据如下：

图 **4-42:** 采集到的日志数据



- 分隔符模式文件

Logtail同时支持分隔符模式的日志解析，示例如下：

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # your config name, must be unique in you k8s cluster
  name: delimiter-file-example
spec:
  # logstore name to upload log
  logstore: k8s-file
  logtailConfig:
    # log file's input type is 'file'
    inputType: file
    configName: delimiter-file-example
    # logtail config name, should be same with [metadata.name]
    inputDetail:
      # 对于分隔符类型的日志，logType设置为delimiter_log
      logType: delimiter_log
      # 日志文件夹
      logPath: /usr/local/ilogtail
      # 文件名，支持通配符，例如log_*.log
      filePattern: delimiter_log.LOG
      # 使用多字符分隔符
      separator: '|&|'
      # 提取的key列表
      key : ['time', 'level', 'method', 'file', 'line', 'message']
      # 用作解析时间的key，如无需求可不填
      timeKey: 'time'
      # 时间解析方式，如无需求可不填
      timeFormat: '%Y-%m-%dT%H:%M:%S'
      # 采集容器内的文件，dockerFile flag设置为true
      dockerFile: true
      # only collect container with "ALIYUN_LOGTAIL_USER_DEFINED_ID
" in docker env config
      dockerIncludeEnv:
```

```
        ALIYUN_LOGTAIL_USER_DEFINED_ID: ''
```

- **JSON模式文件**

  若文件中每行数据为一个JSON object，可以使用JSON方式进行解析，示例如下：

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # your config name, must be unique in you k8s cluster
  name: json-file-example
spec:
  # logstore name to upload log
  logstore: k8s-file
  logtailConfig:
    # log file's input type is 'file'
    inputType: file
    # logtail config name, should be same with [metadata.name]
    configName: json-file-example
    inputDetail:
      # 对于分隔符类型的日志，logType设置为json_log
      logType: json_log
      # 日志文件夹
      logPath: /usr/local/ilogtail
      # 文件名, 支持通配符，例如log_*.log
      filePattern: json_log.LOG
      # 用作解析时间的key，如无需求可不填
      timeKey: 'time'
      # 时间解析方式，如无需求可不填
      timeFormat: '%Y-%m-%dT%H:%M:%S'
      # 采集容器内的文件，dockerFile flag设置为true
      dockerFile: true
      # only collect container with "ALIYUN_LOGTAIL_USER_DEFINED_ID
" in docker env config
      dockerIncludeEnv:
        ALIYUN_LOGTAIL_USER_DEFINED_ID: ""
```

4.8 相关限制说明

表 4-4: 文件采集限制

分类	限制说明
文件编码	支持UTF8/GBK编码日志文件，建议使用UTF8编码以获得更好的处理性能。如果日志文件为其它编码格式则会出现乱码、数据丢失等错误。
日志文件大小	无限制。
日志文件轮转	支持，流转文件名支持配置为`.log*`或者`.log`。
日志解析阻塞时采集行为	日志解析阻塞时，Logtail会将该日志文件FD保持打开状态；若解析阻塞期间出现多次日志文件

分类	限制说明
	轮转，Logtail会尽可能保持各个轮转日志解析顺序。若未解析的日志轮转超过20个，则后续文件不被处理。更多内容请参考相关技术文章。
软链接	支持监控目录为软链接。
单条日志大小	单条日志大小限制为512KB。多行日志按行首正则表达式划分后，每条日志大小限制仍为512KB。若日志超过512KB后，会强制拆分多块进行采集。例如：日志单条1025KB，则第一次处理前512KB，第二次处理512KB，第三次处理1KB。
正则表达式	正则表达式类型支持Perl兼容正则表达式。
同一文件对应多个采集配置	不支持，建议文件采集到一个Logstore，可以配置多份订阅。若有相关需求，可通过为文件配置软连接的方式绕过该限制。
文件打开行为	Logtail会保持被采集文件处于打开状态，若该文件超过5分钟未修改，则会关闭该文件（未发生轮转情况下）。
首次日志采集行为	Logtail只采集增量的日志文件，首次发现文件修改后，若文件大小超过1M，则从最后1M处开始采集，否则从开始位置采集；若配置下发后日志文件一直无修改，则不采集该文件。
非标准文本日志	对于日志中包含'\0'的行，该条日志会被截断到第一个'\0'处。

表 4-5: Checkpoint管理

项目	能力与限制
Checkpoint超时时间	若文件超过30天未修改，则会删除该Checkpoint。
Checkpoint保存策略	定期保存（15分钟），程序退出时会自动保存。
Checkpoint保存位置	保存路径默认为`/tmp/logtail_checkpoint`，可根据 配置启动参数 调整参数。

表 4-6: 配置限制

项目	能力与限制
配置更新	用户的配置更新生效的延时约30秒。
配置动态加载	支持，且其中某一配置更新不影响其他采集。
配置数	理论无限制，建议一台服务器采集配置数不超过100。
多租户隔离	各个采集配置间隔离。详细内容请参考相关技术文章。

表 4-7: 资源、性能限制

项目	能力与限制
日志处理吞吐能力	原始日志流量默认限制为2MB/s（数据会编码压缩后上传，一般压缩率为5-10倍）。超过该日志流量则有可能丢失日志，可根据 配置启动参数 调整参数。
最大性能	单核能力：极简模式日志最大处理能力为100MB/s，正则默认最大处理能力为20MB/s（和正则复杂度有关），分隔符日志最大处理能力为40MB/s，JSON日志最大处理能力为30MB/s；开启多个处理线程性能可提高1.5-3倍左右
监控目录数	主动限制监控的目录层深，避免出现过多消耗用户资源。如果监控上限已到，则放弃监控更多目录和日志文件。限制最多3000个目录（含子目录）。
默认资源限制	默认Logtail最多会占用40%CPU、256MB内存，如日志产生速率较高，可根据 配置启动参数 调整参数。
资源超限处理策略	若3分钟内Logtail占用的相关资源超过最大限制，则Logtail会强制重启，此时数据可能会丢失或重复。

表 4-8: 错误处理限制

项目	能力与限制
网络错误处理	在出现网络异常时会主动重试并自动调整重试间隔。
资源配额超限处理	若数据发送速率超出Logstore最大配额，Logtail会阻塞采集并自动重试。详细内容请参考相关技术文章。。

项目	能力与限制
超时最大尝试时间	若数据持续发送失败超过6小时，则丢弃该数据。
状态自检	支持异常情况下自动重启，例如程序异常退出及使用资源超限等。

表 4-9: 其他限制

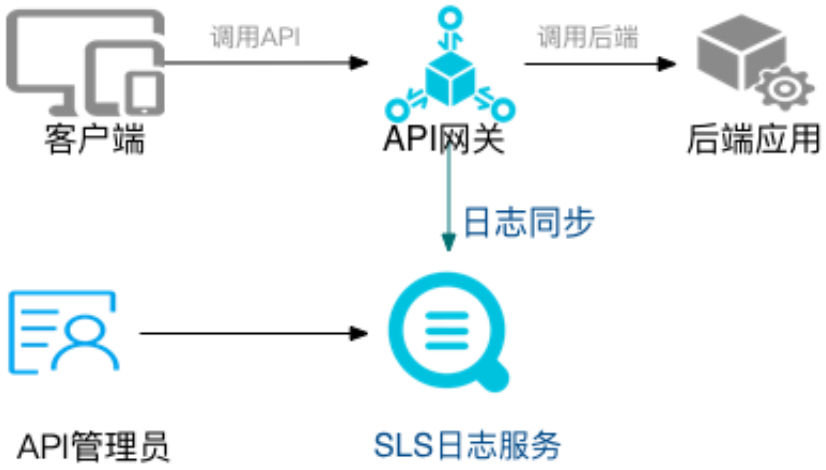
项目	能力与限制
日志采集延迟	正常情况下从日志flush磁盘到Logtail采集改日志延迟不超过1秒（阻塞状态下除外）。
日志上传策略	Logtail会将同一文件的日志自动聚合上传，聚合条件为：日志超过2000条、日志总大小超过2M或者日志采集时间超过3秒，任一条件满足则触发上传行为。

5 云产品采集

5.1 API网关访问日志

阿里云API网关提供API托管服务，在微服务聚合、前后端分离、系统集成上为用户提供诸多便利。访问日志（Acccess Log）是由web服务生成的日志，每一次API请求都对应一条访问记录，内容包括调用者IP、请求的URL、响应延迟、返回状态码、请求和响应字节数等重要信息，可以便于用户了解其Web服务的运行状况。

图 **5-1: API网关**



日志服务支持通过**数据接入向导**采集API网关的访问日志。

功能简介

1. **日志在线查询**。您可以根据日志中任意关键字进行快速的精确或模糊检索，可用于问题定位或者统计查询。
2. **详细调用日志**。您可以检索API调用的详细日志。
3. **自定义分析图表**。您可以根据统计需求将任意日志项自定义统计图表，以满足您日常的业务需要。
4. **预置分析报表**。API网关预定义了一些全局统计图表。包括：请求量大小、成功率、错误率、延时情况、调用API的APP数量，错误情况统计、TOP 分组、TOP API、Top 延迟等等。

字段说明

日志字段	描述
apiGroupUid	API的分组ID
apiGroupName	API分组名称
apiUid	API的ID
apiName	API名称
apiStageUid	API环境ID
apiStageName	API环境名称
httpMethod	调用的HTTP方法
path	请求的PATH
domain	调用的域名
statusCode	HttpStatusCode
errorMessage	错误信息
appId	调用者应用ID
appName	调用者应用名称
clientIp	调用者客户端IP
exception	后端返回的具体错信息
providerAliUid	API提供者帐户ID
region	区域，如：cn-hangzhou
requestHandleTime	请求时间，格林威治时间
requestId	请求ID，全局唯一
requestSize	请求大小，单位：字节
responseSize	返回数据大小，单位：字节
serviceLatency	后端延迟，单位：毫秒

配置步骤

1. 创建Project和Logstore。

   请参考准备工作创建一个Project和Logstore。

   若Logstore已存在请跳过本步骤。

2. 进入数据接入向导。

   创建Logstore后，在**Logstore**列表界面单击数据接入向导图标，进入数据接入向导流程。

3. 选择数据类型

   在云产品日志部分单击**API**网关，并单击下一步，进入**数据源设置**。

4. 设置数据源。

   在**数据源设置**步骤中，检查您是否已完成以下配置：

   a. 是否开通**API**网关服务。

      API 网关为您提供完整的 API 托管服务，辅助用户将能力、服务、数据以 API 的形式开放给合作伙伴，也可以发布到 API 市场供更多的开发者采购使用。

      如您尚未开通**API**网关服务，请按照页面提示开通。

   b. 是否完成**RAM**授权。

      建立分发规则之前需要通过访问控制RAM为日志服务授权，允许日志服务采集您的API网关日志。

      单击右上角的**授权**，完成快捷授权。

   c. 是否已建立分发规则。

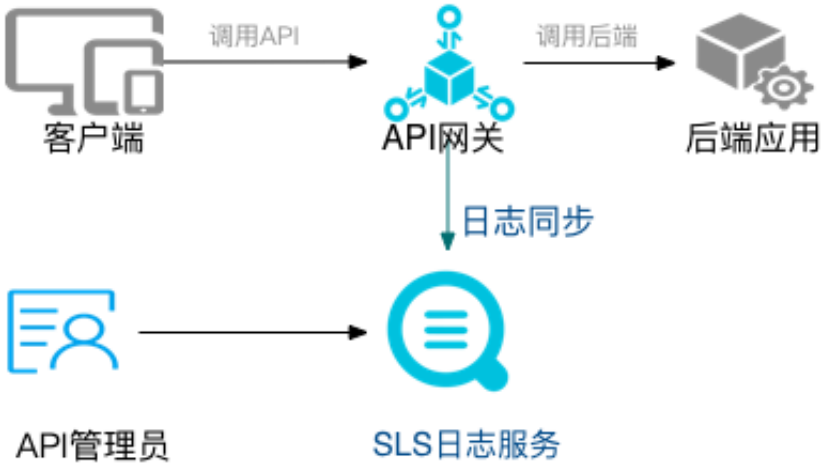
      如您是第一次操作此步骤，系统会自动完成API网关日志导入和分发规则；如您此前已配置过**API**网关日志采集，页面会提示**已经存在日志分发规则**，您可以选择删除旧的分发规则。

   单击下一步，进入**查询分析&可视化**配置。

5. 配置**查询分析&可视化**。

   请按照下图配置索引。索引的配置关系到您的日志检索分析效率，您在仪表盘中也会使用到索引配置，请谨慎修改。

图 5-2: 配置索引



单击下一步，结束配置。日志投递可待有需要时另行配置。

至此，数据接入向导初始化工作完成，您可以选择刚才设置的Logstore api-gateway-access-log进行日志查询、分析，或者进入仪表盘查看报表。

如您需要修改或删除配置，请参考*API*网关文档，在*API*网关控制台操作。

5.2 MNS日志

MNS 的日志管理功能将用户的消息操作日志推送到指定 LoggingBucket 中。用户在控制台上配置将日志推送到日志服务，然后开启该地域队列/主题的日志管理功能，MNS 将自动推送该队列/主题消息的操作日志到指定的 LoggingBucket 中。

- 如果用户将 LoggingBucket 对应的LogService的Project、Logstore删除，或者将授予 MNS 的权限取消，日志将无法正常推送到日志服务。
- 日志延迟时间约5分钟。
- 每个地域配置一个 LoggingBucket，该地域所有开通日志管理功能的队列/主题的消息操作日志均推送到该 LoggingBucket中。
- 每个队列/主题可以独立设置是否开启日志管理功能，默认不开启。

注意事项

- 如果用户将 LoggingBucket 对应的LogService的Project、Logstore删除，或者将授予 MNS 的权限取消，日志将无法正常推送到日志服务。
- 日志延迟时间约5分钟。

- 每个地域配置一个 LoggingBucket，该地域所有开通日志管理功能的队列/主题的消息操作日志均推送到该 LoggingBucket中。
- 每个队列/主题可以独立设置是否开启日志管理功能，默认不开启。

前提条件

1. 您已开通日志服务LogService和消息服务MNS。
2. 您的消息服务日志仅能推送到对应Region下的日志服务Project，请确认您已创建了对应Region的Project和Logstore。
3. 子账号授权请参考子账号授权准备。

操作步骤

1. 开启队列和主题的日志功能。

   此处以开启队列的日志功能为例。

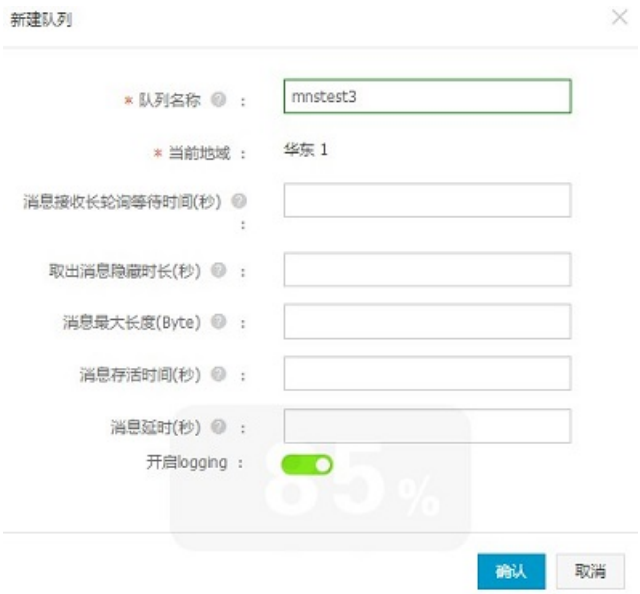
   a. 在消息服务控制台左侧单击**队列**，并选择地域。

   b. 选择需要采集日志的队列，单击**操作**栏中的**修改设置**。

   c. 在修改队列对话框中，打开开启**logging**开关。

   > 说明：
   >
   > - 该功能默认关闭，请确保所有需要采集日志的队列已开启该功能。
   > - 主题的日志功能开启步骤类似，请参考**队列**的操作步骤，打开主题的日志功能。

图 5-3: 开启logging



2. 进入日志管理页面。

   在消息服务控制台单击左侧 **日志管理**，进入日志管理页面。

3. 选择Region。

   在页面上方选择需要推送日志的队列或主题所在的Region，并单击操作列的**配置**。

图 5-4: 选择Region



4. 确认授权与日志服务Project。

   - 如您是首次操作MNS日志推送，请按照页面提示进行快捷授权。
   - 如您没有合适的Project和Logstore，请按照页面提示前往日志服务控制台新建一个Project和Logstore。具体步骤请参考准备工作。

5. 配置推送。

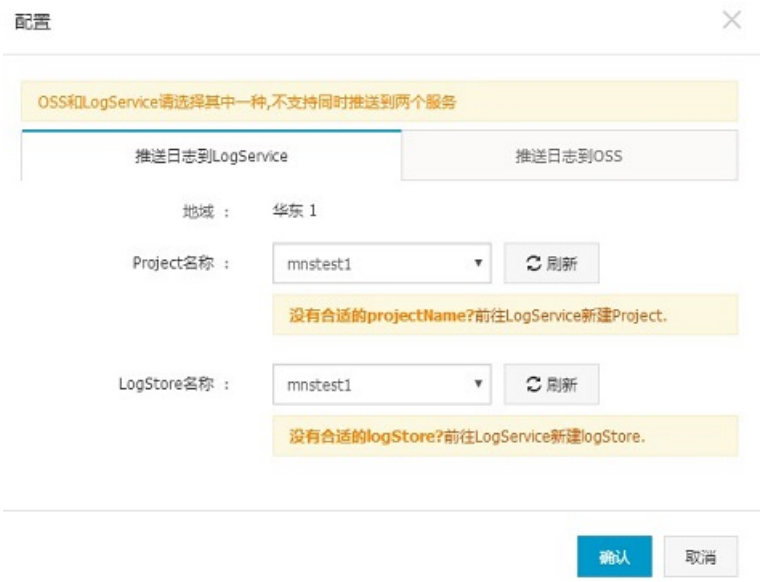
   在**推送日志到LogService**页签中选择对应Region的日志服务Project和Logstore，并单击确认。

说明：

- 请勿取消授权或删除RAM角色，否则会造成MNS日志无法正常推送到日志服务。
- 请保证LoggingBucket和日志服务Project地域一致。
- 创建Project和Logstore后，返回MNS控制台配置页面，单击刷新即可看到新建的Project和Logstore。

配置完成后，单击确认。

图 **5-5:** 配置推送



日志格式

队列消息操作日志

队列消息操作日志是指操作队列消息所产生的日志，比如发送消息、消费消息、删除消息等操作。

一条消息操作日志中包含多个字段，每个字段都有自己的含义。根据操作的不同，消息操作日志所包含的字段也不相同。

日志字段说明

一条消息操作日志中包含多个字段，各个字段的含义如下表所示。

字段	含义
Time	本次操作的发生时间。

字段	含义
MessageId	消息的 MessageId，标识本次操作处理的消息。
QueueName	本次操作对应的队列名称。
AccountId	本次操作对应队列的账号。
RemoteAddress	发起该操作的客户端地址。
NextVisibleTime	该操作执行完成后，这条消息的下次可见时间。
ReceiptHandleInRequest	用户执行该操作时传入的 ReceiptHandle 参数。
ReceiptHandleInResponse	该操作执行完成后，返回给用户的 ReceiptHandle。

各个操作的字段说明

不同操作的日志包含的字段信息各不相同，具体每个操作包含的字段请参考表格。

操作	Time	QueueNa	AccountId	MessageI	RemoteA ess	NextVisib eTime	ReceiptHa dleInResp nse	ReceiptHan dleInReque st
SendMessag e/ BatchSendM essage	有	有	有	有	有	有	-	-
PeekMessag e/ BatchPeekM essage	有	有	有	有	有	-	-	-
ReceiveMes sage/ BatchRecei veMessage	有	有	有	有	有	有	有	-
ChangeMess ageVisibil ity	有	有	有	有	有	有	有	有
DeleteMess age/ BatchDelet	有	有	有	有	有	有	-	有

操作	Time	QueueNa	AccountId	MessageI	RemoteA ess	NextVisib eTime	ReceiptHa dleInResp nse	ReceiptHan dleInReque st
eMessage								

主题消息操作日志

主题消息操作日志是指操作主题消息产生的日志，主要有两类：发布消息和推送消息。

主题消息操作日志各个字段的含义，以及不同的操作所包含的字段信息如下。

日志字段说明

一条消息操作日志中包含多个字段，各个字段的含义如表格所示。

字段	含义
Time	本次操作的发生时间。
MessageId	消息的 MessageId，标识本次操作处理的消息。
TopicName	本次操作对应的主题名称。
SubscriptionName	本次操作对应的订阅名称。
AccountId	本次操作对应主题的账号。
RemoteAddress	发起该操作的客户端地址。
NotifyStatus	MNS 将消息推送给用户时，用户返回的状态码或者相应的错误信息。

各个操作的字段说明

不同操作的日志包含的字段信息各不相同，具体每个操作包含的字段请参考下表。

操作	Time	MessageI	TopicNam	Subscript onName	AccountId	RemoteA ess	NotifyStat us	Subscripti onName
PublishMes sage	有	有	有	-	有	有	-	-
Notify	有	有	有	有	有	-	有	有

NotifyStatus字段

NotifyStatus是推送消息日志特有的字段，可以协助您调查MNS推送消息到Endpoint失败的原因。

根据不同的 NotifyStatus，您可以按照下表建议的处理方法进行处理。

错误码	描述	建议处理方法
2xx	消息推送成功。	-
其它Http状态码	消息推送给用户，Endpoint 返回了非2xx的状态码。	检查 Endpoint 端处理逻辑。
InvalidHost	订阅指定的 Endpoint 不合法。	确认订阅中 Endpiont 是否真实有效，可使用curl/telnet进行确认。
ConnectTimeout	连接订阅指定的 Endpoint 超时。	确认订阅中 Endpoint 当前是否可访问，可使用curl/telnet进行确认。
ConnectFailure	连接订阅指定的 Endpoint 失败。	确认订阅中 Endpoint 当前是否可访问，可使用curl/telnet进行确认。
UnknownError	未知错误。	请工单联系 MNS 技术人员。

5.3 OSS访问日志

日志服务支持采集OSS访问日志，并对采集到的OSS访问日志进行实时查询与分析统计、通过多种可视化方式进行分析结果的直观展示。日志服务对OSS访问日志的专业日志采集分析手段，可以简化您的操作审计和事件回溯，让您的工作更有效率。

前提条件

1. 已开通日志服务。
2. 已开通OSS服务，并成功创建了Bucket。
3. 日志服务Project需要和OSS Bucket在同一个阿里云账号的同一Region下。

配置步骤

1 日志采集授权

单击快捷授权为日志服务授权，授权后日志服务有权限将OSS访问日志分发到您的Logstore中。

若您之前没有授权，也可以在OSS控制台首页单击**日志分析 > 日志采集授权**，根据提示完成授权。

2 关联Bucket

1. 在OSS控制台首页单击**日志分析 > 管理日志服务**，进入日志分析页面。

2. 单击新建关联，进入新建关联步骤。

图 5-6: 新建关联



a. 选择/新建项目。

选择日志服务Project所属的区域、日志服务项目名称，并单击下一步。

> 说明：
>
> - 您的日志服务**Project**与**OSS Bucket**须处于同一区域，多个**Bucket**日志可以采集至同一个**Logstore**中。
> - 如您之前没有在当前区域中创建对应的日志服务Project，您需要输入Project名称新建一个。

b. 选择日志库。

选择对应的Logstore名称，并单击下一步。

> 说明：
> 如您之前没有在当前区域中创建对应的日志服务Logstore，您需要输入Logstore名称新建一个。

c. 关联**Bucket**。

选择对应的Bucket名称，并单击**提交**。您可以选择将当前区域下的多个Bucket与该Logstore关联。

图 5-7: 关联Bucket



您已成功建立分发规则。请根据页面提示，前往日志服务控制台设置索引信息。

3 配置查询分析

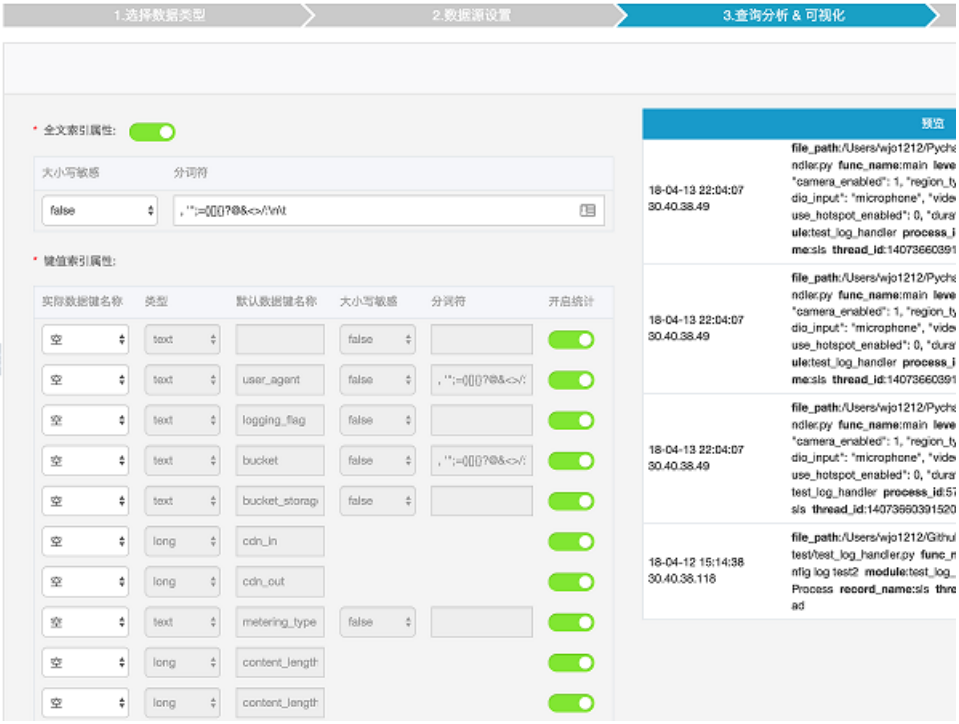
1. 关联**Bucket**配置完成后，根据页面提示，单击弹出对话框中的设置索引，跳转到日志服务控制台查询分析 **&** 可视化页面。

   图 **5-8:** 配置查询分析



2. 日志服务已预设了OSS日志所需的查询索引，字段说明请查看OSS日志格式。确认后单击下一步。

图 **5-9:** 配置索引



说明：

系统默认为您关联了Bucekt的Logstore创建4个特定的仪表盘，配置完成后您可以在**仪表盘**页面查看。您也可以在OSS控制台的**日志分析**页面中单击对应日志库**分析日志**，并单击左侧的仪表盘名称，直接查看仪表盘。

图 **5-10:** 分析日志



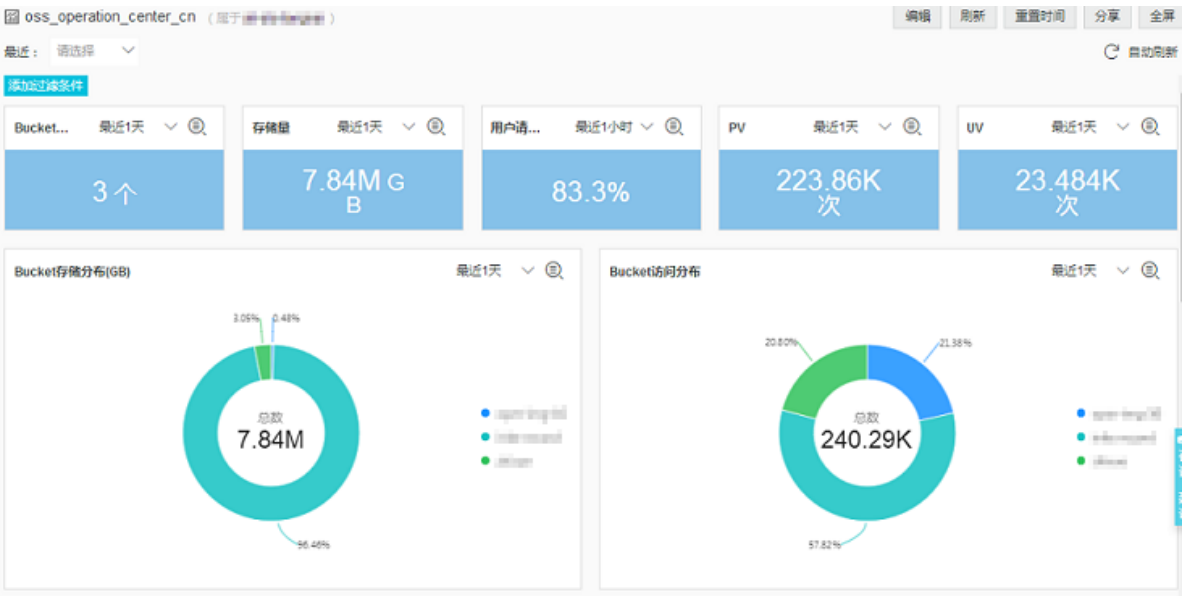
3. 您可以按需要配置日志投递、ETL，或者直接单击**确认**，完成配置。

默认仪表盘

日志服务提供了4个开箱即用的报表中心：

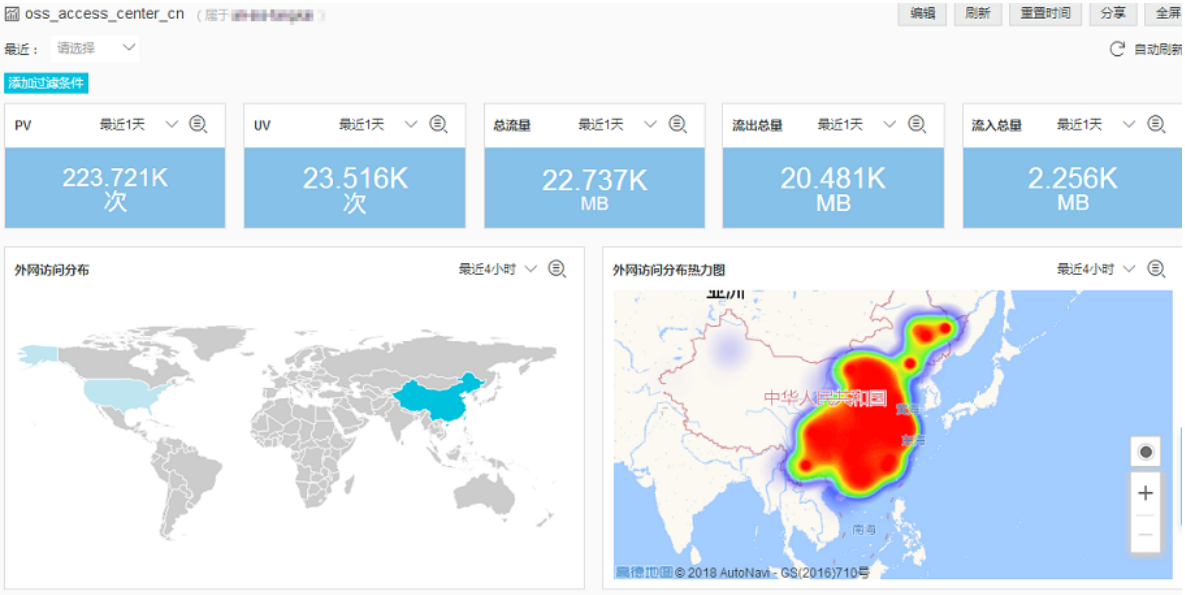
- **oss_operation_center**，展示总体运营状况的信息

图 5-11: oss_operation_center



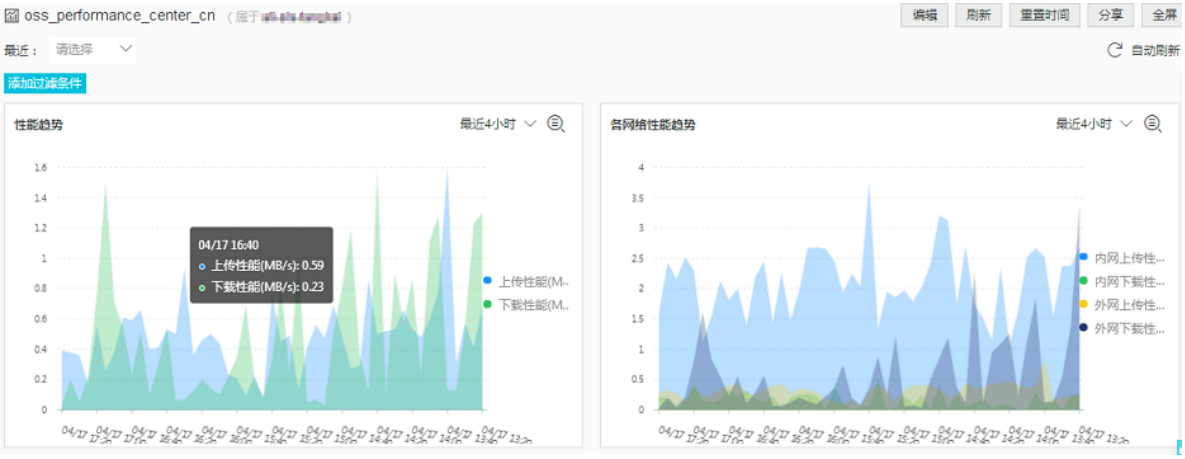
- **oss_access_center**，展示针对访问日志的统计信息

图 5-12: oss_access_center



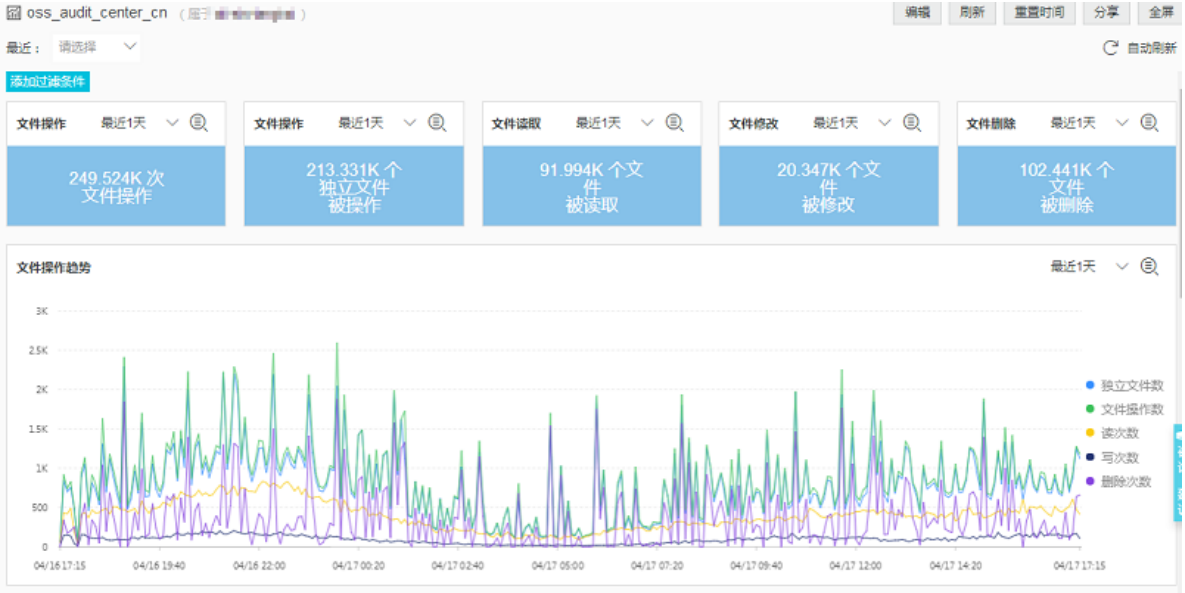
- **oss_performance_center**，展示针对性能的统计信息

图 5-13: oss_performance_center



- **oss_audit_center**，展示文件删除修改的统计信息

图 5-14: oss_audit_center



更多信息请参考视频。

OSS日志格式

表 5-1: OSS日志格式

日志类型	说明
访问日志	记录了对应的Bucket的所有访问日志。实时收集。
批量删除日志	记录了批量删除的具体删除信息。实时收集。

日志类型	说明
	说明： 当用户调用DeleteObjects时，访问日志中会有一条请求记录。但因为删除的文件信息存在用户请求的HTTP Body中，访问日志中的object会是-，要参考具体的删除文件的列表，需要查看批量删除日志，可以通过request_id关联。
每小时计量日志	记录特定Bucket每个小时累计的一些统计计量，延迟为几小时，用于辅助分析。

访问日志

表 5-2: 访问日志

字段	含义	示例
__topic__	主题，固定为oss_access_log。	-
time	访问时间，即OSS收到请求的时间，如果需要时间戳可以使用域__time__。	27/Feb/2018:13:58:45
owner_id	Bucket拥有者阿里云ID。	12345678
User-Agent	HTTP的User-Agent头。	curl/7.15.5
logging_flag	是否开启logging，即是否开启了日志定期导出到OSS Bucket的功能。	true
bucket	Bucket名称。	bucket123
content_length_in	请求头中Content-Length的值（字节）。	12345
content_length_out	回应头中Content-Length的值（字节）。	12345
object	用户请求的object，URL编码，查询时可以使用select url_decode(object)解码。	data%2Fcur_file.txt
object_size	对象大小，即对应请求对象的大小，单位为Bytes。	1234

字段	含义	示例
operation	访问类型，完整访问类型列表与描述，请参考访问类型。	GetObject
bucket_location	Bucket所在集群，一般格式是`oss-<region>-id`。	oss-cn-beijing-f
request_uri	用户请求的URI，包括query-string，路径是URL编码，查询时可以使用`select url_decode(request_uri)`解码。	/1518085703067732%2Fcur_file.txt HTTP/1.1
error_code	OSS返回的错误码，完整错误码列表与描述，请参考*OSS*错误响应。	NoSuchKey
request_length	HTTP请求的大小，包括header。单位为字节。	376
client_ip	请求发起的IP地址，即客户端IP、其网络防火墙或者Proxy的IP。	1.2.3.4
response_body_length	HTTP响应中body的大小，即HTTP response的body的大小，不包括header。	123
http_method	HTTP请求方法。	GET
referer	请求的HTTP Referer。	http://www.abc.com
requester_id	请求者阿里云ID，匿名访问为“-”。	12345678
request_id	请求ID，表示本次请求的ID，用于OSS技术人员问题排查。	5A7C39674857FB9FFFFFF
response_time	请求响应时间，单位为毫秒。	123
server_cost_time	OSS服务器处理时间，即OSS服务器处理本次请求所花的时间，单位为毫秒。	123
http_type	HTTP请求类型，即http或https请求。	http

字段	含义	示例
sign_type	签名类型，完整签名类型列表和描述，请参考签名类型。	NormalSign
http_status	HTTP状态，即OSS请求返回的HTTP状态。	200
sync_request	同步请求类型，完整同步请求类型列表和描述，请参考同步请求类型。	cn
bucket_storage_type	Bucket存储类型，完整Bucket存储类型列表和描述，请参考Bucket存储类型。	standard
host	请求访问域名。	bucket123.oss-cn-beijing.aliyuncs.com
vpc_addr	用户访问oss的域名对应的vip的地址，整形格式，用于OSS技术人员问题排查。	1234567890
vpc_id	用户通过vpc访问时的vpc id，用于OSS技术人员问题排查。	1234
delta_data_size	object大小的变化量，若没有变化为0；如果不是上传请求，则为“-”。	280

批量删除日志

当用户调用DeleteObjects时，访问日志中会有一条请求记录。但因为删除的文件信息存在用户请求的HTTP Body中，访问日志中的`object`会是`-`。参考具体的删除文件的列表，需要查看批量删除日志。批量删除日志的字段及说明如下，可以通过`request_id`关联。

表 5-3: 批量删除日志

字段	说明	示例
__topic__	主题，固定为`oss_batch_delete_log`。	-

字段	说明	示例
client_ip	请求发起的IP地址，客户端IP或者其网络防火墙或者Proxy的IP。	1.2.3.4
user_agent	User-Agent，HTTP的User-Agent头。	curl/7.15.5
bucket	Bucket名称。	bucket123
error_code	OSS返回的错误码，完整错误码列表与描述，请参考*OSS*错误响应。	NoSuchKey
request_length	request的大小，HTTP请求的大小，包括header。单位为字节。	376
response_body_length	response的body的大小，HTTP response的body的大小，不包括header。	123
object	用户请求的object，URL编码，查询时可以使用`select url_decode(object)`解码。	data%2Fcur_file.txt
object_size	对象大小，对应请求对象的大小，单位Bytes。	1234
operation	访问类型，完整访问类型列表与描述，请参考访问类型 。	GetObject
bucket_location	Bucket所在集群，一般格式是`oss-<region>-id`。	oss-cn-beijing-f
http_method	HTTP请求方法。	POST
referer	请求的HTTP Referer。	http://www.abc.com
request_id	请求ID，表示本次请求的ID，用于OSS技术人员问题排查。	5A7C39674857FB9FFFFFF
http_status	HTTP状态，OSS请求返回的HTTP状态。	200

字段	说明	示例
sync_request	同步请求类型，完整同步请求类型列表和描述，请参考同步请求类型。	cdn
request_uri	用户请求的URI，包括query-string, 路径是URL编码，查询时可以使用`select url_decode(request_uri)`解码。	/1518085703067732%2Fcur_file.txt HTTP/1.1
host	请求访问域名。	bucket123.oss-cn-beijing.aliyuncs.com
logging_flag	是否开启logging，即是否开启了原来的日志定期导出功能。	true
server_cost_time	OSS服务器处理时间，单位为毫秒。	123
owner_id	Bucket拥有者阿里云ID。	12345678
requester_id	请求者阿里云ID，匿名访问为“-”。	12345678
delta_data_size	object大小的变化量，若没有变化为0；如果不是上传请求，则为“-”。	280

每小时计量日志

记录特定Bucket每个小时累计的一些统计计量，几小时延迟，仅仅辅助分析参考用。

表 5-4: 每小时计量日志

字段	说明	示例
__topic__	主题，固定为`oss_metering_log`。	-
owner_id	Bucket拥有者阿里云ID。	12345678
bucket	Bucket名称。	bucket123
cdn_in	CDN流入量，单位为字节。	123
cdn_out	CDN流出量，单位为字节。	123

字段	说明	示例
get_request	GET请求次数。	123
intranet_in	内网流入量，单位为字节。	123
intranet_out	内网流出量，单位为字节。	123
network_in	外网流入量，单位为字节。	123
network_out	外网流出量，单位为字节。	123
put_request	PUT请求次数。	123
storage_type	Bucket存储类型，完整Bucket存储类型列表和描述，请参考*Bucket*存储类型。	standard
storage	Bucket存储量，单位为字节。	123
metering_datasize	非标准存储的计量数据大小。	123
process_img_size	处理的图像大小，单位为字节。	123
process_img	处理图像。	123
sync_in	同步流入量，单位为字节。	123
sync_out	同步流出量，单位为字节。	123
start_time	计量开始时间戳。	1518084000
end_time	计量截止时间戳，一般一小时一个计量时间单位。	1518087600
region	Bucket所在区域。	cn-beijing
bucket_location	Bucket所在集群，一般格式是`oss-<region>-id`。	oss-cn-beijing-f

访问类型

表 **5-5:** 访问类型

操作值	描述
AbortMultiPartUpload	断点上传-中止
AppendObject	追加上传文件
CommitTransition	CommitTransition

操作值	描述
CompleteUploadPart	完成断点上传
CopyObject	复制文件
DeleteBucket	删除Bucket
DeleteLiveChannel	删除LiveChannel
DeleteObject	删除文件
DeleteObjects	删除多个文件
ExpireObject	ExpireObject
GetBucket	列举文件
GetBucketAcl	获取Bucket权限
GetBucketCors	查看Bucekt的CORS规则
GetBucketEventNotification	获取Bucekt通知配置
GetBucketInfo	查看Bucket信息
GetBucketLifecycle	查看Bucket的Lifecycle配置
GetBucketLocation	查看Bucket区域
GetBucketLog	查看Bucket访问日志配置
GetBucketReferer	查看Bucket防盗链设置
GetBucketReplication	查看跨区域复制
GetBucketReplicationProgress	查看跨区域复制进度
GetBucketStat	获取bucket的相关信息
GetBucketWebSite	查看Bucket的静态网站托管状态
GetLiveChannelStat	获取LiveChannel状态信息
GetObject	读取文件
GetObjectAcl	获取文件访问权限
GetObjectInfo	获取文件信息
GetObjectMeta	查看文件信息
GetObjectSymlink	获取symlink文件的详细信息
GetPartData	获取断点文件块数据

操作值	描述
GetPartInfo	获取断点文件块信息
GetProcessConfiguration	获取Bucekt图片处理配置
GetService	列举Bucket
HeadBucket	查看Bucket信息
HeadObject	查看文件信息
InitiateMultipartUpload	初始化断点上传文件
ListMultiPartUploads	列举断点事件
ListParts	列举断点块状态
Options	Options
PostObject	表单上传文件
PostProcessTask	提交相关的数据处理，比如截图等
PostVodPlaylist	创建LiveChannel点播列表
ProcessImage	图片处理
PutBucket	创建Bucket
PutBucketCors	设置Bucket的CORS规则
PutBucketLifecycle	设置Bucket的Lifecycle配置
PutBucketLog	设置Bucket访问日志
PutBucketWebSite	设置Bucket静态网站托管模式
PutLiveChannel	创建LiveChannel
PutLiveChannelStatus	设置LiveChannel状态
PutObject	上传文件
PutObjectAcl	修改文件访问权限
PutObjectSymlink	创建symlink文件
RedirectBucket	bucket endpoint重定向
RestoreObject	解冻文件
UploadPart	断点上传文件
UploadPartCopy	复制文件块

操作值	描述
get_image_exif	获取图片的exif信息
get_image_info	获取图片的长宽等信息
get_image_infoexif	获取图片的长宽以及exif信息
get_style	获取Bucket样式
list_style	列举Bucket的样式
put_style	创建Bucket样式

关于每个操作的具体信息，请参考*API*概览。

同步请求类型

表 **5-6:** 同步请求类型

同步请求类型	描述
-	一般请求
cdn	CDN回源

签名类型

表 **5-7:** 签名类型

签名类型	描述
NotSign	未签名
NormalSign	一般方式签名
UriSign	通过URL签名
AdminSign	管理员账号

关于签名的进一步信息，请参考用户签名验证。

Bucket存储类型

表 5-8: Bucket存储类型

存储类型	描述
standard	标准存储类型
archive	归档存储类型
infrequent_access	低频访问存储类型

关于每个存储类型的进一步信息，请参考存储类型介绍。

5.4 DDoS高防日志

5.4.1 简介

阿里云DDoS高防IP是针对互联网服务器（包括非阿里云主机）在遭受大流量的DDoS攻击后导致服务不可用的风险，推出的付费增值服务，用户可以通过配置高防IP，将攻击流量引流到高防IP，确保源站的稳定可靠。

背景信息

互联网界的安全一直都不断的面临着挑战，以DDoS攻击为代表的网络威胁直接对网络安全产生严重的影响。

DDoS攻击正在朝着大规模、移动化、全球化的方向发展。据近年来的调查报告显示，DDoS攻击的频率呈现出增长的趋势。黑客攻击的隐蔽性强，能够控制大量的安全措施差的云服务商、IDC、甚至海量摄像头发起攻击，其攻击已经形成了成熟的黑色产业链，并且越来越有组织化。同时，攻击的方式向两极化发展，慢速攻击、混合攻击尤其是CC攻击占比不断增大，这给检测防御造成更大的难度。一方面，超过1Tbps的攻击峰值屡见不鲜、100G攻击次数成倍增长，另一方面，应用层攻击也在大幅度翻倍

根据卡巴斯基*2018Q1*的*DDoS*风险报告，中国依然主要的DDoS攻击源和目标。 主要被攻击的行业是互联网、游戏、软件公司、金融等。超过80%DDoS攻击会混合HTTP攻击，而CC攻击尤其隐蔽，因此通过日志对访问和攻击行为进行即时分析研究、附加防护策略就显得尤其重要。

日志服务支持实时采集阿里云*DDoS*高防*IP*的网站访问日志、CC攻击日志，并支持对采集到的日志数据进行实时检索与分析，并以仪表盘形式展示查询结果。

功能优势

- **配置简单**：轻松配置即可采集实时高防日志。
- **实时分析**：依托日志服务，提供实时日志分析，并提供开箱即用的报表中心，对CC攻击状况以及客户访问细节了如指掌。
- **实时告警**：支持基于特定指标定制准实时的监测与告警，确保关键业务异常时可及时响应。
- **生态体系**：支持对接其他生态如流计算、云存储、可视化方案，进一步挖掘数据价值。
- **免费额度**：提供特定免费数据导入额度，以及免费3天的日志存储、查询与实时分析。并可自由扩展存储时间，以便合规管理、溯源、备案等。支持不限时长的存储，存储成本低至0.35元/GB/月。

限制与说明

- **专属日志库不支持写入其他数据**。

  专属日志库用于存入DDoS高防IP的网站日志，因此**不支持写入其他数据**。其他查询、统计、报警、流式消费等功能没有限制。

- **按量计费。没有为任何网站开启DDoS高防日志采集功能则不收费**。

  DDoS高防日志功能按照日志服务的收费项进行计费，没有为任何网站开启DDoS高防日志采集功能则不收费。日志服务为**按量计费**模式，并提供一定的**免费额度**。详细计费说明请参考费用说明。

应用场景

- **排查网站访问异常**

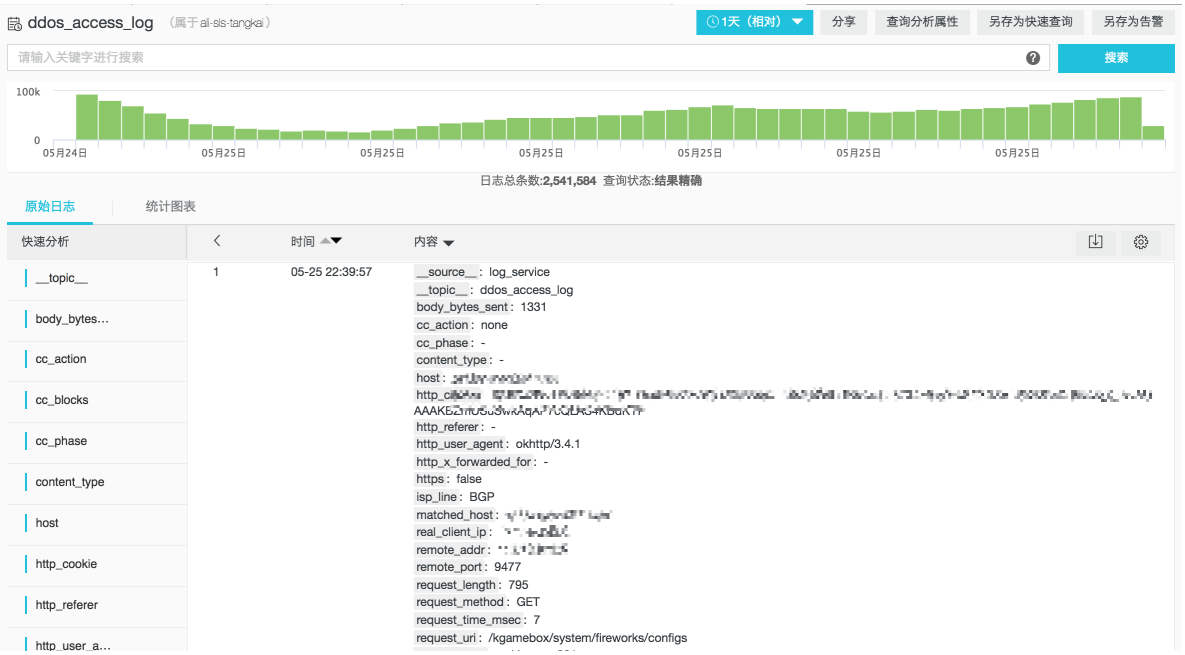
  配置日志服务采集DDoS高防日志后，您可以对采集到的日志进行实时查询与分析。使用SQL语句分析DDoS访问日志，可以对网站的访问异常进行快速排查和问题分析，并查看读写延时、运营商分布等信息。

  例如，通过以下语句查看DDoS访问日志：

  ```
  __topic__: ddos_access_log
  ```

  查询结果：

图 5-15: DDoS访问日志



- 追踪CC攻击者来源

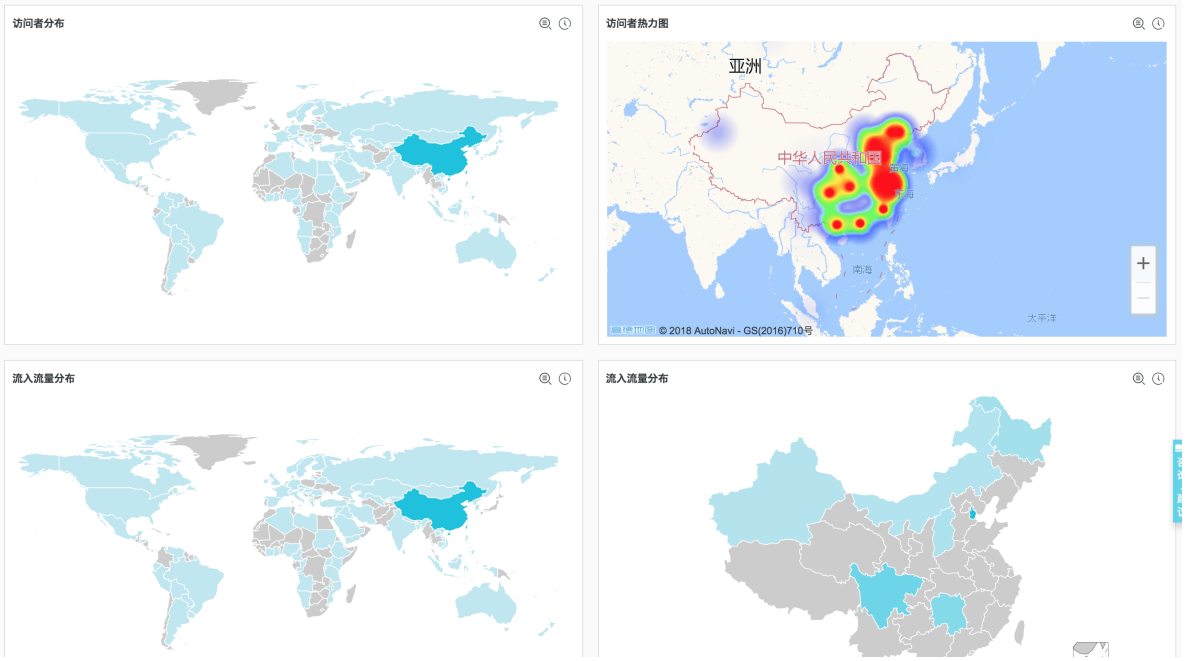
  DDoS访问日志中记录了CC攻击者的分布及来源，通过对DDoS访问日志进行实时查询与分析，您可以对CC攻击者进行来源追踪、溯源攻击事件，为您的应对策略提供参考。

  例如，通过以下语句分析DDoS访问日志中记录的CC攻击者国家分布：

```
__topic__: ddos_access_log and cc_blocks > 0| SELECT ip_to_country
(if(real_client_ip='-', remote_addr, real_client_ip)) as country,
count(1) as "攻击次数" group by country
```

  将分析结果用仪表盘表示：

图 5-16: CC攻击者来源



- 例如，通过以下语句查看访问PV：

```
__topic__: ddos_access_log | select count(1) as PV
```

将分析结果用仪表盘表示：

图 5-17: 访问PV



- 网站运营分析

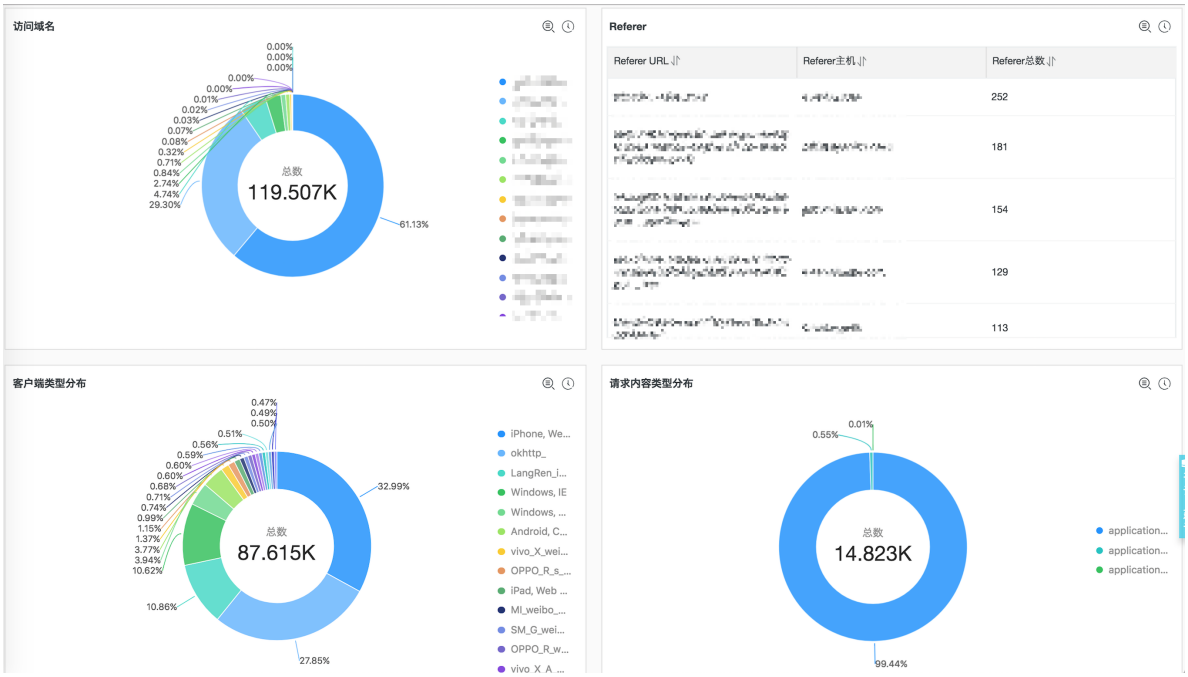
DDoS访问日志中实时记录网站访问数据，您可以对采集到的访问日志数据进行SQL查询分析，得到实时的访问情况，例如判断网站热门程度、访问来源及渠道、客户端分布等，并以此辅助网站运营分析。

例如，查看来自各个网络云因上的访问者流量分布：

```
__topic__: ddos_access_log | select ip_to_provider(if(real_clien
t_ip='-', remote_addr, real_client_ip)) as provider, round(sum(
request_length)/1024.0/1024.0, 3) as mb_in group by provider having
 ip_to_provider(if(real_client_ip='-', remote_addr, real_client_ip))
 <> '' order by mb_in desc limit 10
```

将分析结果用仪表盘展示：

图 5-18: 访问客户端分布



5.4.2 采集步骤

您可以在DDoS高防IP控制台为网站开启DDos高防日志采集功能。

前提条件

1. 开通DDos高防IP产品，购买DDoS高防实例，并配置上线。
2. 开通日志服务产品。

背景信息

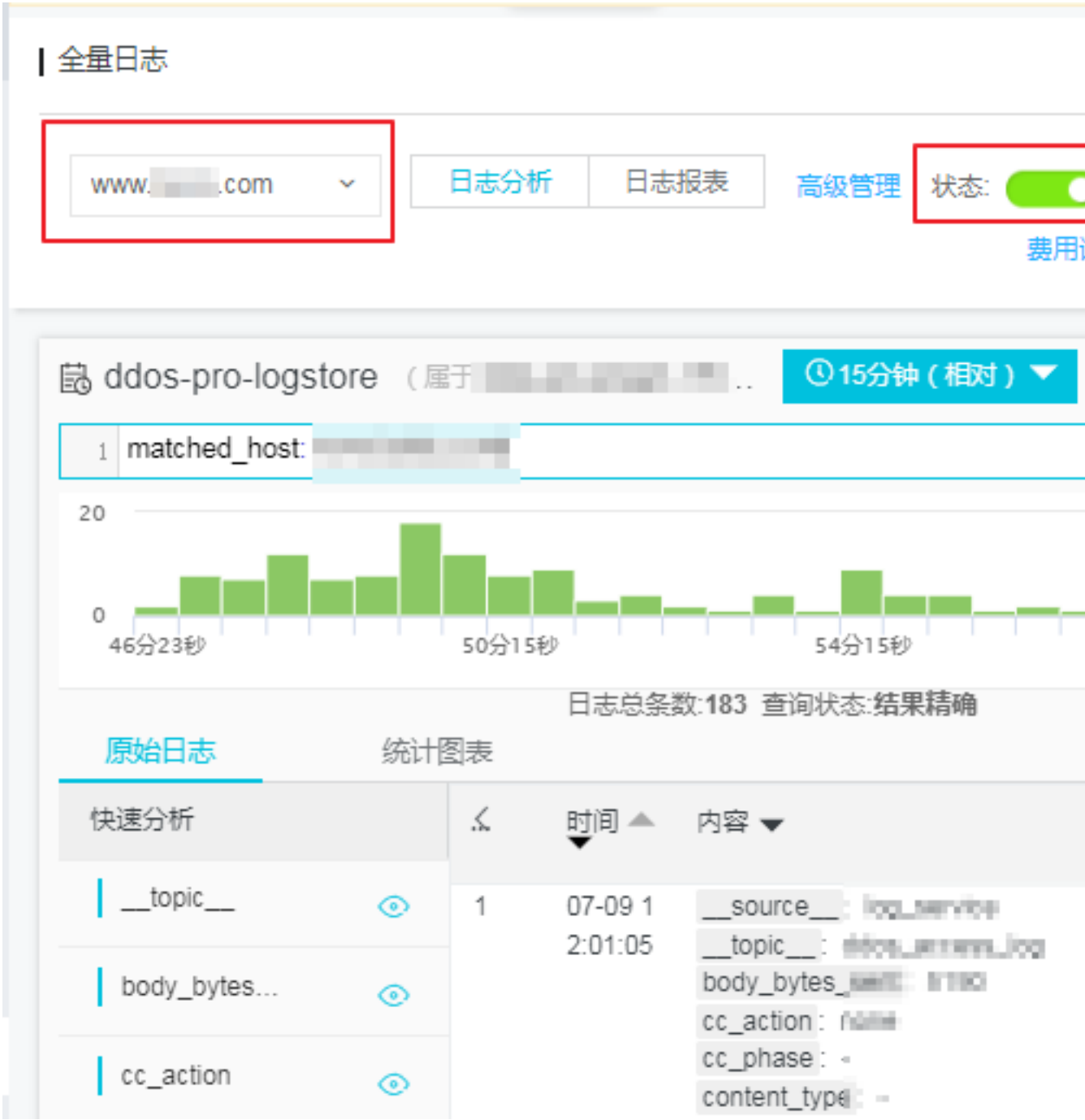
日志服务支持实时采集阿里云**DDoS**高防**IP**的网站访问日志、CC攻击日志，并支持对采集到的日志数据进行实时检索与分析，并以仪表盘形式展示查询结果，通过日志对访问和攻击行为进行即时分析研究、协助安全部门制定防护策略。

操作步骤

1. 登录DDoS高防IP控制台，在左侧导航栏中选择**日志 > 全量日志**，进入**全量日志**页面。
2. 若您是第一次配置DDoS高防日志采集功能，请根据页面提示授权。

   授权后DDoS有权限将DDos高防日志分发到您的Logstore中。
3. 选择您需要开启DDoS高防日志采集功能的网站，并开启右侧的**状态**开关。

图 **5-19:** 开启功能



至此，您已成功为当前网站开启了DDoS高防日志采集功能。日志服务会在您的账号下为您自动创建一个专属日志库和专属Logstore，DDoS会将所有开启了该功能的网站高防日志导入到这个专属日志库中。专属日志库和专属Logstore等默认配置请参见默认配置。

表 **5-9:** 默认配置

默认配置项	配置内容
Project	默认为您创建Project`ddos-pro-logstore`。
Logstore	默认为您创建Logstore。Logstore名称由您购买的DDoS的所在地域决定。 • 大陆地域的DDoS实例：`ddos-pro-project-`阿里云账户*`ID`*`-cn-hangzhou` • 其他地域的DDoS实例：`ddos-pro-project-`阿里云账户*`ID`*`-ap-southeast-1` DDoS高防IP日志采集功能产生的所有日志都会保存在这个Logstore中。
地域	• DDoS地域为中国大陆地区的，默认Project默认保存在杭州地域。 • DDoS地域为其他地区的，默认Project默认保存在新加坡地域。
Shard	默认为您创建2个Shard，并开启自动分裂*Shard* 功能。
日志存储时间	默认保存3天，在免费额度之内。3天以上的日志自动删除。若您需要更长的存储时间，可以自定义修改。详细步骤请参考费用说明中的**如何修改网站日志的存储时间**部分。
仪表盘	默认为您创建两个仪表盘，分别为： • **`ddos-pro-logstore`**`_ddos_operation_center`：运营中心 • **`ddos-pro-logstore`**`_ddos_access_center`：访问中心 关于仪表盘的更多信息，请参考*DDoS*高防日志-日志报表。

您可以在当前的**全量日志**页面对采集到的日志进行实时查询与分析等操作，日志字段说明请查看下图。另外，日志服务为您创建了DDoS运营中心和访问中心两个仪表盘，您也可以自定义配置仪表盘。

字段	说明	示例
__topic__	日志主题（Topic），固定为ddos_access_log。	-

字段	说明	示例
body_bytes_sent	请求发送Body的大小，单位为字节。	2
content_type	内容类型。	application/x-www-form-urlencoded
host	源网站。	api.zhihu.com
http_cookie	请求cookie。	k1=v1;k2=v2
http_referer	请求referer，若没有，显示为-。	http://xyz.com
http_user_agent	请求User Agent。	Dalvik/2.1.0 (Linux; U; Android 7.0; EDI-AL10 Build/HUAWEIEDISON-AL10)
http_x_forwarded_for	通过代理跳转的上游用户IP。	-
https	该请求是否为HTTPS请求，其中： • true：该请求是HTTPS请求。 • false：该请求是HTTP请求。	true
matched_host	匹配的配置的源站，可能是泛域名。未匹配则为-。	*.zhihu.com
real_client_ip	访问客户的真实IP，获取不到时为-。	1.2.3.4
isp_line	线路信息，例如BGP、电信、联通等。	电信
remote_addr	请求连接的客户端IP。	1.2.3.4
remote_port	请求连接的客户端端口号。	23713
request_length	请求长度，单位为字节。	123
request_method	请求的HTTP方法。	GET
request_time_msec	请求时间，单位为微秒。	44
request_uri	请求路径。	/answers/377971214/banner

字段	说明	示例
server_name	匹配到的host名，没有匹配到则为`default`。	api.abc.com
status	HTTP状态。	200
time	时间。	2018-05-02T16:03:59+08:00
cc_action	CC防护策略行为，例如none、challenge、pass、close、captcha、wait、login、n等。	close
cc_blocks	表示CC防护是否阻止，其中： • 1表示阻止。 • 其他内容表示通过。	1
cc_phase	CC防护策略，包括seccookie、server_ip_blacklist、static_whitelist、 server_header_blacklist、server_cookie_blacklist、server_args_blacklist、qps_overmax等。	server_ip_blacklist
ua_browser	浏览器。	ie9
ua_browser_family	浏览器系列。	internet explorer
ua_browser_type	浏览器类型。	web_browser
ua_browser_version	浏览器版本。	9.0
ua_device_type	客户端设备类型。	computer
ua_os	客户端操作系统。	windows_7
ua_os_family	客户端操作系统系列。	windows
upstream_addr	回源地址列表，格式为`IP:Port`，多个地址用逗号分隔。	1.2.3.4:443
upstream_ip	实际回源地址IP。	1.2.3.4
upstream_response_time	回源响应时间，单位为秒。	0.044
upstream_status	回源请求HTTP状态。	200

字段	说明	示例
user_id	阿里云AliUID。	12345678

后续操作

- 单击日志分析，查询分析采集到的日志数据。
- 单击日志报表，查看内置仪表盘。
- 单击高级管理，跳转到日志服务控制台，对您采集到的日志数据进行查询、统计、流式消费、设置告警等多样化处理。

5.4.3 日志分析

DDoS高防IP产品在控制台的全量日志页面嵌入了日志服务日志分析和日志报表页面。您对于特定网站开通了DDoS高防日志功能之后，可以在当前页面对采集到的日志数据进行实时查询与分析、查看或编辑仪表盘、设置监控告警等。

操作步骤

1. 登录DDoS高防IP控制台，在左侧导航栏中选择日志 > 全量日志，进入全量日志页面。
2. 选择您需要开启DDoS高防日志采集功能的网站，确认右侧的状态为开启。
3. 单击日志分析。

   当前页面内嵌了日志服务的查询分析页面，系统会自动为您输入查询语句，如`matched_host: www.aliyun.com`，查看基于选定网站的日志数据。

图 **5-20:** 日志分析



4. 输入您的查询分析语句，选择日志时间范围后单击**查询**。

> **说明：**
>
> 您的DDoS高防日志的默认保存时间为3天，3天之前的日志数据会被删除。默认情况下只能查询到过去三天内的日志数据。若您需要修改日志保存时间，请参考修改日志保存时间。

图 5-21: 查询日志



基于查询分析页面，您还可以对查询到的日志数据进行以下操作：

- **自定义查询与分析**

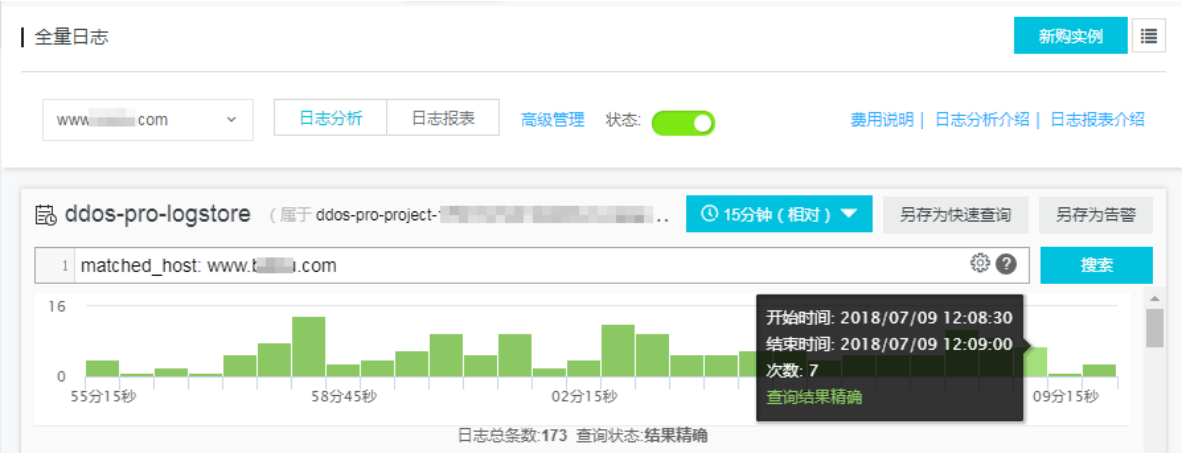
  日志服务定义了一系列查询语法和分析语法，支持多种复杂场景下的日志查询。详情请参考自定义查询与分析。

- **查看日志的时间分布**

  搜索框下方展示了符合查询时间和查询语句的日志的时间分布，以时间为横轴、数量为纵轴的柱状图形式展示。并显示查询到的日志总数。

> **说明：**
> 可以在柱状图上滑动以选择更小范围的时间区域，时间选择器会自动更新为选择的时间范围，并刷新结果。

图 **5-22:** 日志的时间分布



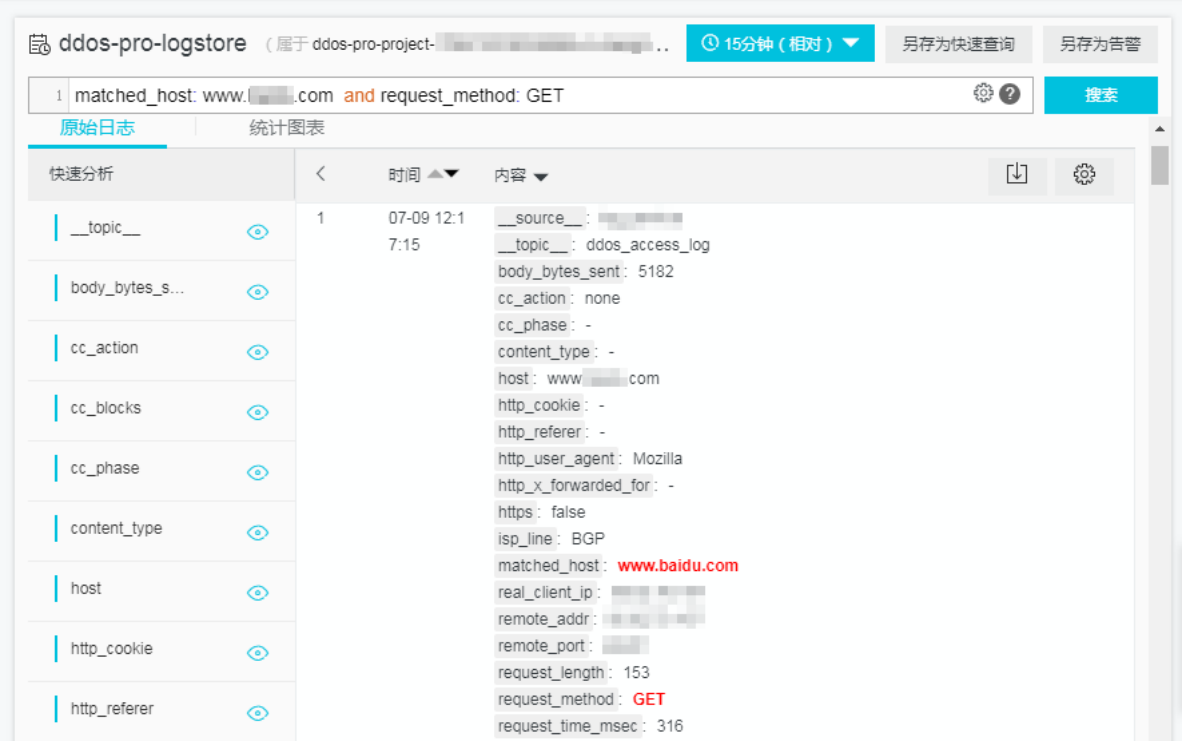
- 查看原始日志

  原始日志页签中，以分页的形式展示了每一条日志的详细内容，包括时间、内容以及其中的各个字段。您可以对列进行排序、对当前查询结果进行下载，也可以单击齿轮按钮，选择特定的字段进行展示等。

在页面中点击相应字段的值或分词，搜索框中自动输入相应的搜索条件。例如鼠标单击`request_method: GET`中的值`GET`，会自动给搜索框加入如下语句：

```
原来的搜索语句 and request_method:  GET
```

图 **5-23:** 原始日志



- 查看分析图表

  日志服务支持图表形式展示分析结果，您可以在统计图表页面根据需要选择不同的图表类型。详情请参考分析图表。

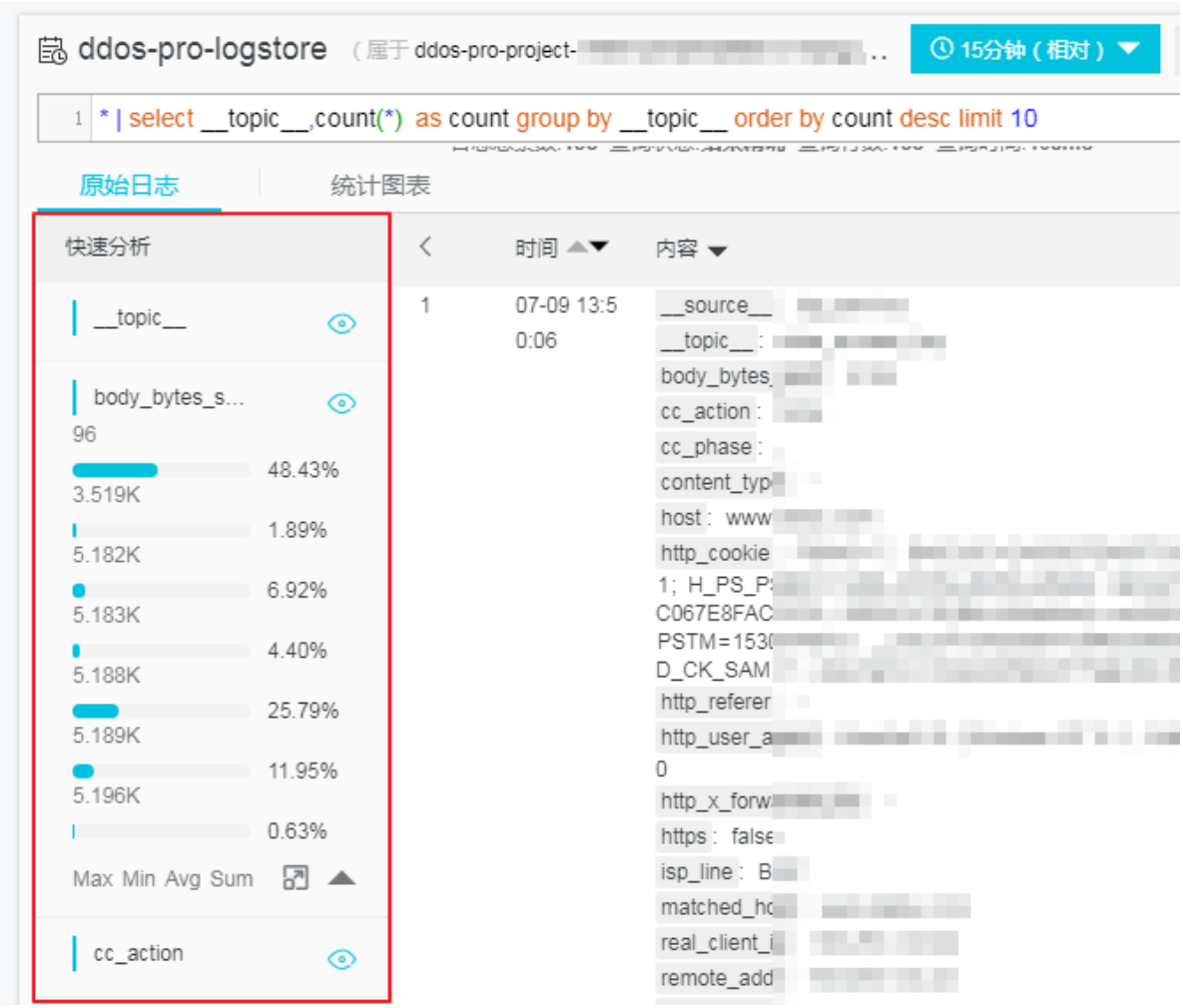
图 **5-24:** 统计图表



- 快速分析

  快速分析功能为您提供一键交互式查询体验，帮助您快速分析某一字段在一段时间内的分布情况，减少索引关键数据的时间成本。详细说明请参考快速分析

图 **5-25:** 快速分析



自定义查询分析

日志查询语句由查询语法（Search）和分析语法（Analytics）两个部分组成，中间通过|进行分割：

```
$Search | $Analytics
```

类型	说明
查询（Search）	查询条件，可以由关键词、模糊、数值等、区间范围和组合条件等产生。如果为空或*，则代表所有数据。
分析（Analytics）	对查询结果或全量数据进行计算和统计。

说明：

两部分均为可选，当Search部分为空时代表针对该时间段所有数据不过滤任何条件，直接对结果进行统计。当Analysis部分为空时，代表只返回查询结果，不做统计。

查询语法

日志服务查询语法支持**全文查询**和**字段查询**，查询框支持换行显示、语法高亮等功能。

- **全文查询**

  不需要指定字段，直接输入关键字查询。可以用双引号（""）包裹关键字，多个关键字之间以空格或`and`分割。

  示例：

  - **多关键字查询**

    搜索包含所有`www.aliyun.com`和`error`的日志。例如：

    ```
    www.aliyun.com error
    ```

    或者：

    ```
    www.aliyun.com and error
    ```

  - **条件查询**

    这里搜索所有包含`www.aliyun.com`并且包含`error`或者`404`的日志。例如：

    ```
    www.aliyun.com and (error or 404)
    ```

  - **前缀查询**

    这里搜索所有包含`www.aliyun.com`并且包含`failed_`开头关键字。例如：

    ```
    www.aliyun.com and failed_*
    ```

    说明：

    查询只支持后缀加`*`，不支持前缀`*`，如：`*_error`。

- **字段查询**

  日志服务支持基于字段进行更精准的查询。

  可实现数值类型字段的比较，格式为`字段 : 值`或`字段 >= 值`，通过`and`、`or`等进行组合。也可以和全文搜索组合使用，同样通过`and`、`or`组合。

DDoS网站访问日志和攻击日志同样可以基于字段查询，各个字段的含义、类型、格式等信息请查看*DDoS*日志字段。

示例：

- **查询多字段**

  搜索所有`www.aliyun.com`被CC攻击的日志：

  ```
  matched_host: www.aliyun.com and cc_blocks: 1
  ```

  如果要搜索某个客户端`1.2.3.4`对网站`www.aliyun.com`的所有错误404的访问日志，可以这样：

  ```
  real_client_ip: 1.2.3.4 and matched_host: www.aliyun.com and
  status: 404
  ```

  说明：

  示例中用的字段`matched_host`、`cc_blocks`、`real_client_ip`和`status`等都是DDoS访问与攻击日志的字段，详细的字段列表和信息，可以参考*DDoS*日志字段。

- **查询数值字段**

  搜索所有响应时间超过5秒的慢请求日志：

  ```
  request_time_msec > 5000
  ```

  也支持区间查询，查询响应时间大于5秒且小于等于10秒的日志：

  ```
  request_time_msec in (5000 10000]
  ```

  该查询还可以通过以下语句实现：

  ```
  request_time_msec > 5000 and request_time_msec <= 10000
  ```

- **查询日字是否存在**

  针对特定字段是否存在进行查询：

  - 查询存在`ua_browser`字段的日志：`ua_browser: *`
  - 查询不存在`ua_browser`字段的日志：`not ua_browser: *`

详细的查询语法说明请参考索引与查询。

分析语法

您可以使用SQL/92语法对日志数据进行分析与统计，日志服务支持的语法与函数请查看实时分析简介。



说明：

- 分析语句中可以省略SQL标准语法中的`from 表格名`语句，即`from log`。
- 日志数据默认返回前100条，您可以通过*LIMIT*语法修改返回范围。

基于日志时间的查询分析

每一条DDoS日志都有一个字段`time`表示日志的时间，格式为`年-月-日T时:分:秒+时区`。例如`2018-05-31T20:11:58+08:00`，其中时区为`UTC+8`区，也就是北京时间。同时，每条日志都有一个内置字段：`__time__`，也表示这条日志的时间，以便在统计时进行基于时间的计算。其格式为*Unix*时间戳，本质是一个自从1970-1-1 0:0:0 UTC时间开始的累计过去的秒数。因此实际使用时，经过可选的计算后，需要格式化才可以展示。

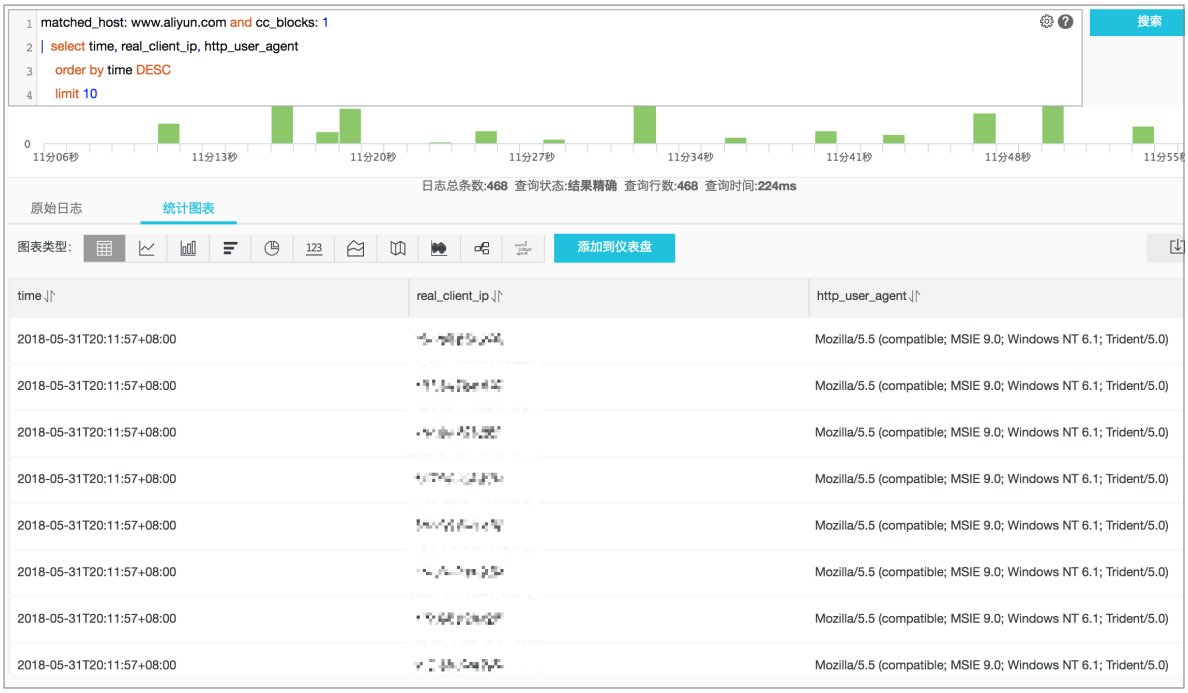
- **选择并展示时间**

  这里在特定时间范围内，选择网站`www.aliyun.com`被CC攻击的最新10条日志，展示其中时间、来源IP以及访问客户端，直接使用字段`time`：

```
matched_host: www.aliyun.com and cc_blocks: 1
|  select time, real_client_ip, http_user_agent
    order by time desc
```

```
    limit 10
```

图 **5-26:** 选择并展示时间



- 计算时间

  查询CC攻击过后的天数，可以使用__time__进行计算：

```
matched_host: www.aliyun.com  and cc_blocks: 1
|  select time,
          round((to_unixtime(now()) - __time__)/86400, 1) as "
days_passed",              real_client_ip, http_user_agent
      order by time desc
      limit 10
```



说明：

这里使用round((to_unixtime(now()) - __time__)/86400, 1)，先用to_unixtime将now()获取的时间转化为Unix时间戳，再与内置时间字段__time__相减，获得已经过去的时间秒数。最后除以86400，即一天的总秒数，再用函数round(data, 1)圆整为小数点后1位数的值，可得出每条攻击日志距离现在已经过去了几天。

图 5-27: 查询结果

time	days_passed	real_client_ip	http_user_agent
2018-05-31T20:11:57+08:00	26.6	[redacted]	Mozilla/5.5 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
2018-05-31T20:11:57+08:00	26.6	[redacted]	Mozilla/5.5 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
2018-05-31T20:11:57+08:00	26.6	[redacted]	Mozilla/5.5 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
2018-05-31T20:11:57+08:00	26.6	[redacted]	Mozilla/5.5 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
2018-05-31T20:11:57+08:00	26.6	[redacted]	Mozilla/5.5 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)

- 基于特定时间分组统计

如果想知道特定时间范围内，某个网站每天被CC攻击的趋势如何，使用如下SQL：

```
matched_host: www.aliyun.com  and cc_blocks: 1
| select date_trunc('day', __time__) as dt,
         count(1) as PV
      group by dt
   order by dt
```



说明：

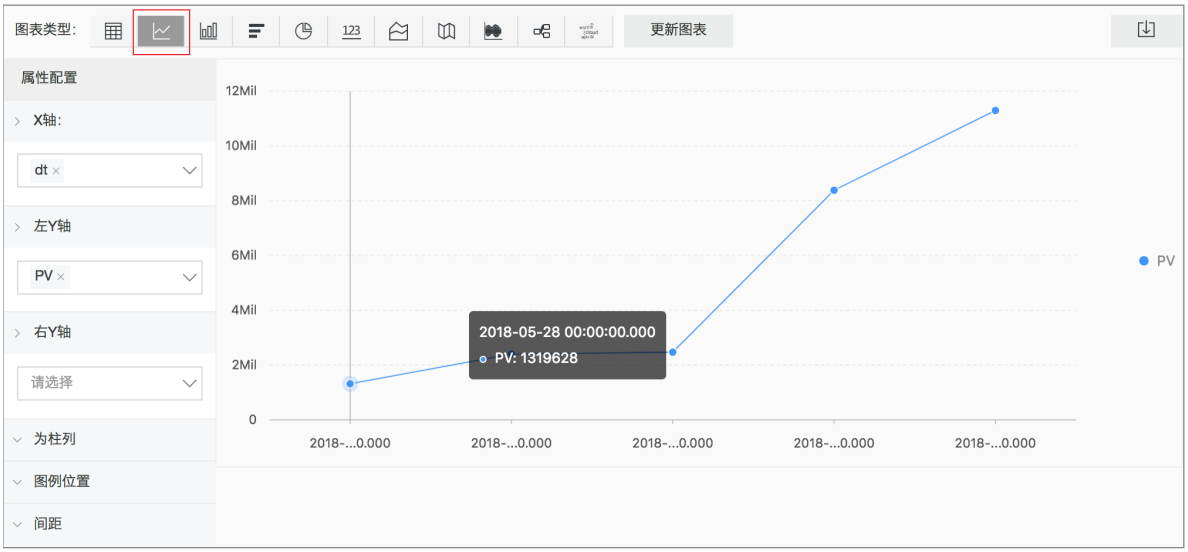
这里使用内置时间字段__time__，传给函数date_trunc('day', ..)进行时间按天对齐，将每条日志分组到了其所属的天的分区中进行统计总数（count(1)），并按照分区时间块排序。函数date_trunc第一个参数提供更多其他单位进行对齐，包括second、miniute、hour、week、month、year等，函数说明请参考日期和时间函数。

图 5-28: 统计结果

dt	PV
2018-05-28 00:00:00.000	1319628
2018-05-29 00:00:00.000	2402020
2018-05-30 00:00:00.000	2473332
2018-05-31 00:00:00.000	8381076
2018-06-01 00:00:00.000	11293642

折线图方式展示：

图 5-29: 折线图



- 基于时间分组统计

如果想知道更灵活的分组下时间规律，例如某个网站每5分钟被CC攻击的趋势，需要进行数学计算。可以使用如下SQL：

```
matched_host: www.aliyun.com  and cc_blocks: 1
| select from_unixtime(__time__ - __time__% 300) as dt,
         count(1) as PV
      group by dt
   order by dt
   limit 1000
```

说明：

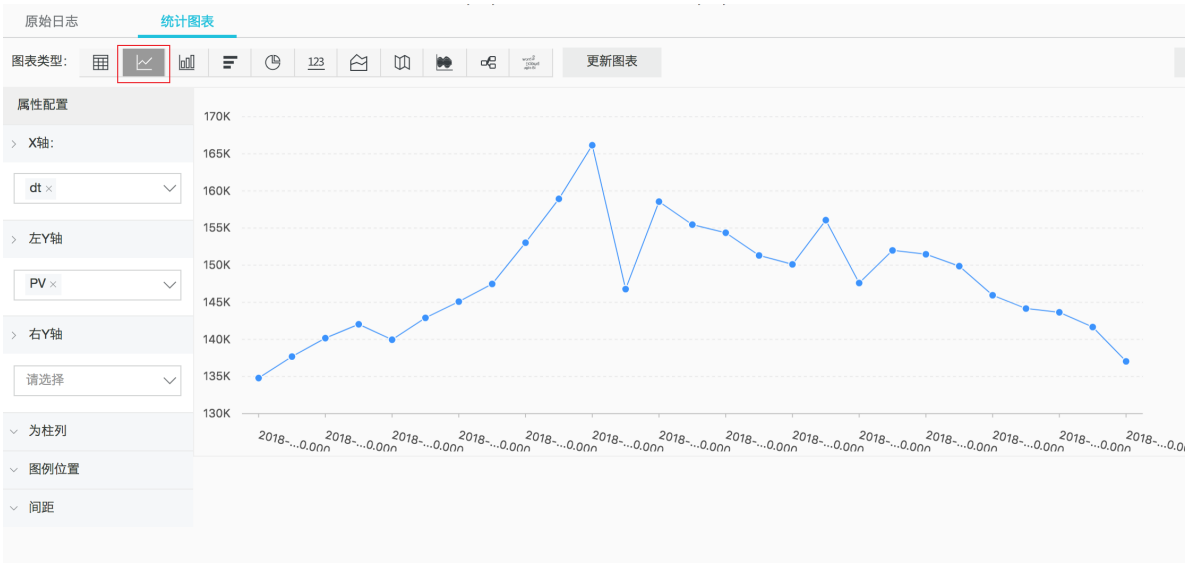
使用计算的内置时间字段计算`__time__ - __time__% 300`，同时使用函数`from_unixtime`进行格式化，将每条日志分组到了一个5分钟（300秒）的分区中进行统计总数（count(1)），并按照分区时间块排序，获得前1000条，相当于选择时间内的前83小时的数据。

图 **5-30:** 时间分组统计结果

dt	PV
2018-05-31 21:30:00.000	134795
2018-05-31 21:35:00.000	137691
2018-05-31 21:40:00.000	140171
2018-05-31 21:45:00.000	142037
2018-05-31 21:50:00.000	139958
2018-05-31 21:55:00.000	142906
2018-05-31 22:00:00.000	145093
2018-05-31 22:05:00.000	147474

折线图方式展示：

图 **5-31:** 折线图



更多关于时间解析的函数，例如将一个时间格式转化为另外一个格式，需要使用`date_parse`与`date_format`，函数说明请参考日期和时间函数。

基于客户端IP的查询分析

DDoS日志中有反映真实客户端IP的字段`real_client_ip`，但用户通过代理并跳转头中IP有误等情况下无法拿到用户真实IP时，可以直接使用直连客户端IP的字段`remote_addr`来代替。

- **攻击者国家分布**

  这里对某个网站进行CC攻击的来源国家分布：

```
matched_host: www.aliyun.com  and cc_blocks: 1
| SELECT ip_to_country(if(real_client_ip='-', remote_addr,
real_client_ip)) as country,
         count(1) as "攻击次数"
   group by country
```

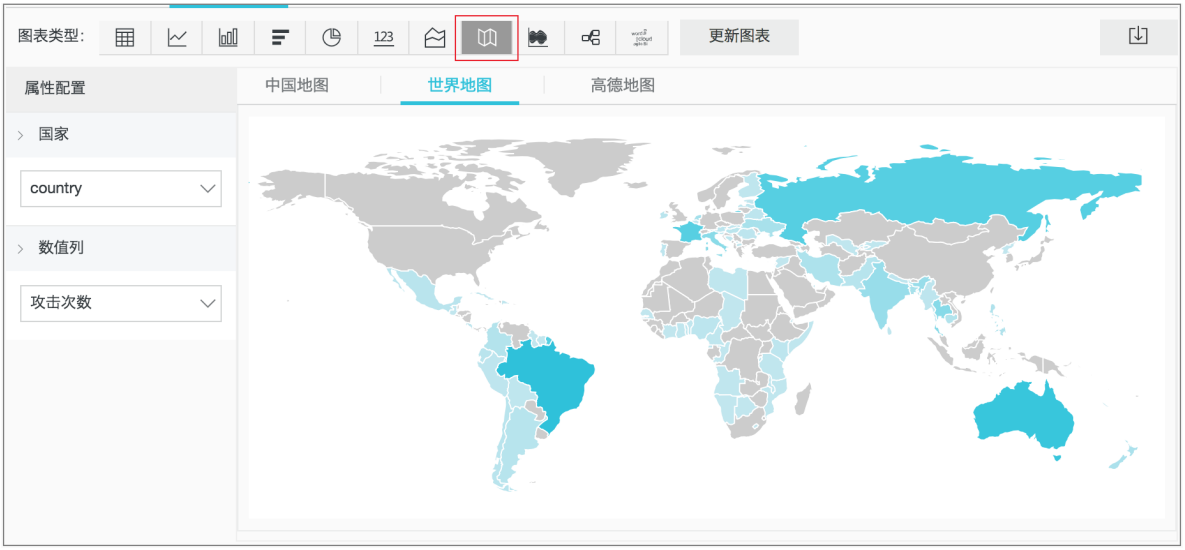
> 说明：
> 这里先用函数`if(condition, option1, option2)`来选择字段`real_client_ip`或者`real_client_ip`（当`real_client_ip`为`-`时）。然后将获得的IP传给函数`ip_to_country`得到这个IP对应的国家信息。

图 **5-32:** 攻击者国家分布-分析结果

country	攻击次数
菲律宾	6321
斯洛文尼亚	521
吉布提	91
多哥	9
印度	14436
爱沙尼亚	65
莱索托	12

世界地图方式展示：

图 5-33: 世界地图



- 访问者省份分布

如果您希望获得更详细的基于省份的分布，可以使用函数ip_to_province，例如：

```
matched_host: www.aliyun.com  and cc_blocks: 1
| SELECT ip_to_province(if(real_client_ip='-', remote_addr,
real_client_ip)) as province,
         count(1) as "攻击次数"
   group by province
```

说明：

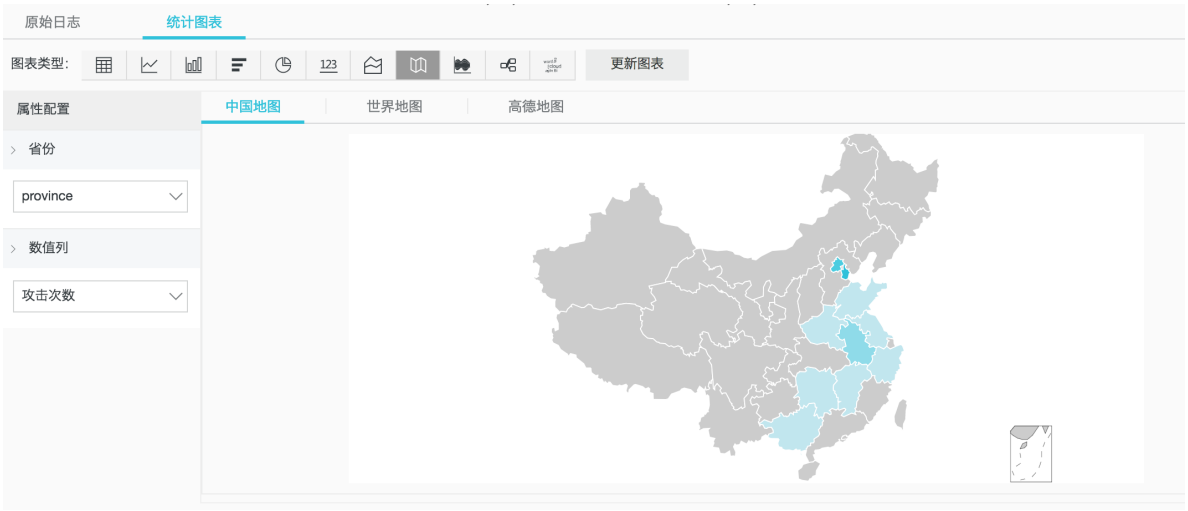
这里使用了另外一个IP函数ip_to_province来获得一个IP的所属省份。如果是中国以外的IP地址，依然会尝试转化为其国家所属省份（州），但在选择中国地图展示时，会无法展示出来。

图 5-34: 访问者省份分布-分析结果

province	攻击次数
江苏省	53
湖南省	2
北京市	509026
河南省	1411
安徽省	205629
广西壮族自治区	503
天津市	723121
浙江省	318

中国地图方式展示：

图 5-35: 中国地图



- 攻击者热力分布

如果期望获得一张攻击者的热力图，可以使用另外一个函数ip_to_geo，例如：

```
matched_host: www.aliyun.com  and cc_blocks: 1
| SELECT ip_to_geo(if(real_client_ip='-', remote_addr, real_clien
t_ip)) as geo,
         count(1) as "攻击次数"
   group by geo
```

```
    limit 10000
```

说明：

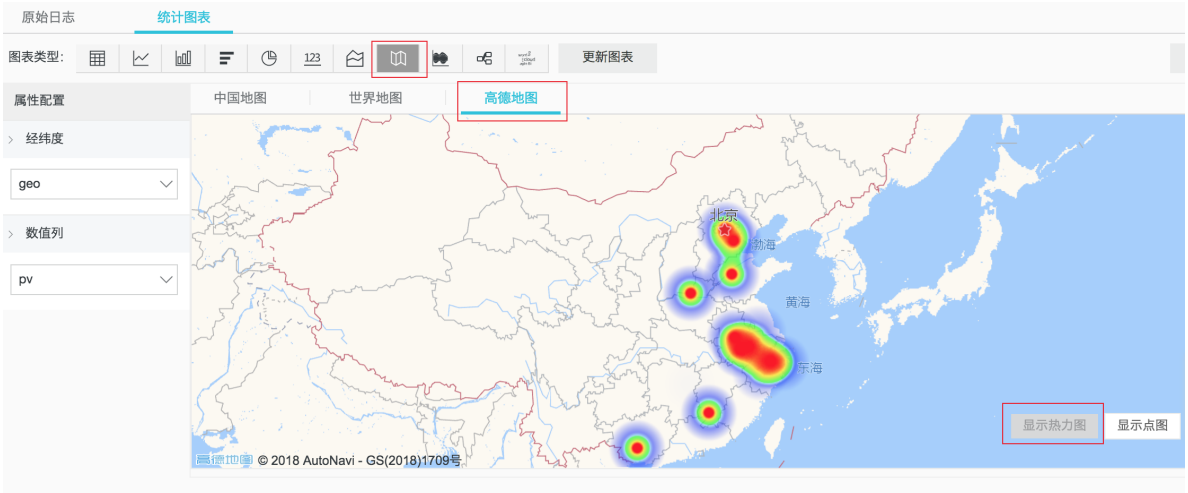
这里使用了另外一个IP函数`ip_to_geo`来获得一个IP的所在经纬度，并获取前1万条。

图 5-36: 攻击者热力分布-分析结果

geo	pv
31.8639,117.281	81378
36.6683,116.997	656
30.0135,120.288660	72
39.1422,117.177	723121
31.1461,118.571	124143
22.8167,108.316670	503
25.85,114.933	673
32.2109,119.455	53

高德地图方式展示：

图 5-37: 高德地图



基于IP的更多解析功能，例如获得IP所属运营商`ip_to_provider`、判断IP是内网还是外网`ip_to_domain`等，可以参考*IP*地理函数。

5.4.4 日志报表

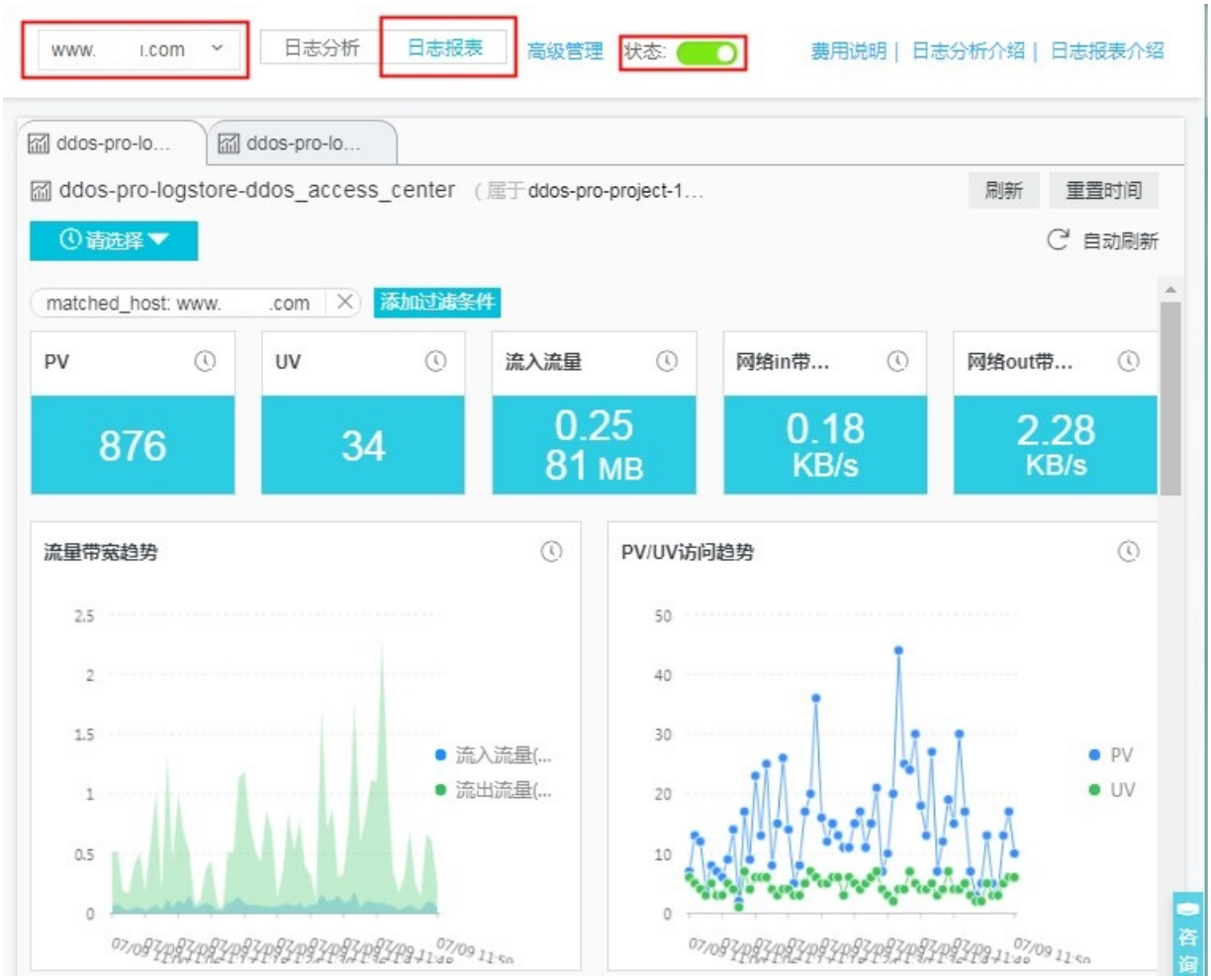
日志报表页面内嵌了日志服务的仪表盘页面。该页面为您展示您的默认仪表盘，您可以在当前页面通过修改时间范围、添加过滤条件等操作，查看多种筛选条件下的仪表盘数据。

查看报表

1. 登录DDoS高防IP控制台，在左侧导航栏中选择日志 > 全量日志，进入全量日志页面。
2. 选择您需要开启DDoS高防日志采集功能的网站，确认右侧的状态为开启。
3. 单击日志报表。

   当前页面内嵌了日志服务的仪表盘页面，系统会自动为您添加过滤条件，如`matched_host: www.aliyun.com`，查看基于选定网站的日志报表。

图 5-38: 查看报表



为网站开启DDoS高防日志采集功能之后，日志服务为您自动创建两个默认的仪报表，即运营中心和访问中心，关于默认仪表盘的更多信息，请查看默认仪表盘。

仪表盘	仪表盘名称	说明
ddos-pro-logstore_ ddos_operation_center	DDoS运营中心	展示被DDoS保护的网站目前的总体运营状况，包括有效请求状况、流量、趋势以及被CC攻击的流量、峰值、攻击者分布等数据。
ddos-pro-logstore_ ddos_access_center	DDoS访问中心	展示被DDoS保护的网站目前的总体被访问状况，包括PV/UV趋势与带宽峰值、访问者分布、流量线路分布、客户端类型分布、请求分布、被访问网站分布等数据。

图 **5-39:** 默认仪表盘



除了查看报表之外，您还可以进行以下操作：

- 选择时间范围

- 添加或编辑过滤条件
- 查看图表

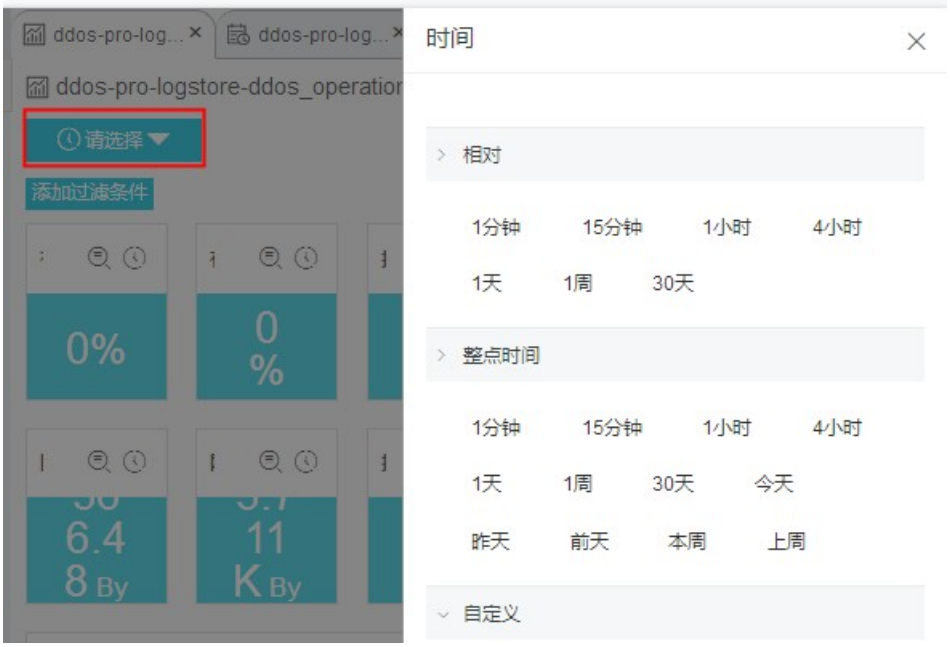
时间选择器

仪表盘页面的所有图表都是基于不同时间段的数据统计结果，例如访问量的默认时间范围为1天，访问趋势为30天。如您想要设置当前页面的所有图表均按照同样的时间范围显示，可以设置时间选择器。

1. 单击请选择。
2. 在弹出的设置框中选择您的设置。您可以选择相对时间、整点时间或设置自定义时间。

> 说明：
>
> - 修改时间范围后，所有图表的时间都会改成这个时间范围。
> - 时间选择器仅在当前页面提供临时的图表查看方式，系统不保存该设置。您下次查看报表时，系统仍会为您展示默认的时间范围。

图 5-40: 设置时间范围



过滤条件

当您选择网站并单击日志报表，进入仪表盘页面后，系统会自动为您添加过滤条件，如 `matched_host: www.aliyun.com`，查看基于选定网站的日志报表。

您可以通过设置过滤条件修改报表的数据展示范围。

- 查看所有网站的全局报表

  清空过滤条件，可以对当前日志库ddos-pro-logstore进行全局的报表展示。

- 添加更多过滤条件

  您可以设置**key**和**value**，进一步对报表数据进行筛选。多个过滤条件之间为AND关系。

  例如查看通过电信线路访问请求的总体情况：

  图 **5-41:** 添加过滤条件



说明：

字段isp_line是DDoS日志的字段，表示连接入口的运营商网络，关于更详细的完整字段列表和信息，可以参考*DDoS*日志字段。

图表类型

报表展示区域按照预定义的布局展示多个报表，包括如下几个类型，更多图表类型请参考图表说明。

图表类型	说明
数字	表示一些重要指标，如有效请求率、攻击峰值等。
线/面积图	表示一些重要指标特定时间单元内的趋势图，如流入带宽趋势、攻击拦截趋势等。

图表类型	说明
地图	表示一些访问者、攻击者的地理分布，如CC攻击者国家分布、访问热点分布等。
饼图	表示一些信息的分布，例如被攻击网站前10、客户端类型分布等。
表格	展示攻击者列表等信息，一般分多个列。
地图	表示一些数据的地理分布。

默认仪表盘

- 运营中心

  运营中心展示被DDoS保护的网站目前的总体运营状况，包括有效请求状况、流量、趋势以及被CC攻击的流量、峰值、攻击者分布等。

图表	类型	默认时间范围	描述	样例
有效请求包率	单值	1小时（相对）	有效请求，即非CC攻击或400错误的请求个数在所有请求总数的占比。	95%
有效请求流量率	单值	1小时（相对）	有效请求在所有请求总流量的占比。	95%
接收流量	单值	1小时（相对）	有效请求流入流量总和，单位MB。	300 MB
攻击流量	单值	1小时（相对）	CC攻击的流入流量总和，单位MB。	30 MB
流出流量	单值	1小时（相对）	有效请求流出流量总和，单位MB。	300 MB
网络in带宽峰值	单值	1小时（相对）	网站请求的流入流量速率的最高	100 Bytes/s

图表	类型	默认时间范围	描述	样例
			峰值，单位为Bytes/s。	
网络out带宽峰值	单值	1小时（相对）	网站请求的流出流量速率的最高峰值，单位为Bytes/s。	100 Bytes/s
接收数据包	单值	1小时（相对）	有效请求（非CC攻击）的流入请求个数，单位为个。	30000 个
攻击数据包	单值	1小时（相对）	CC攻击的请求个数总和，单位为个。	100 个
攻击峰值	单值	1小时（相对）	CC攻击的最高峰值，单位为个/峰值。	100 个/分钟
流入带宽与攻击趋势	双线图	1小时（整点）	每分钟的有效请求和攻击请求的流量带宽的趋势图。单位为KB/s。	-
请求与拦截趋势	双线图	1小时（整点）	每分钟的请求和拦截的CC攻击请求总数的趋势图。单位为个/分钟。	-
有效请求率趋势	双线图	1小时（整点）	每分钟的有效请求（非CC攻击或400错误的请求）个数在所有请求总数的占比趋势图。	-
访问状态分布趋势	流图	1小时（整点）	每分钟的各种请求处理状态（400、304、20	-

图表	类型	默认时间范围	描述	样例
			等）的趋势图单位为个/分钟。	
CC攻击者分布	世界地图	1小时（相对）	CC攻击的次数总和在来源国家的分布。	-
CC攻击者分布	中国地图	1小时（相对）	CC攻击的次数总和在来源省份（中国）的分布。	-
攻击者列表	表格	1小时（相对）	前100个攻击最多的攻击者信息，包括IP、地域城市、网络、攻击次数和攻击总流量。	-
攻击接入线路分布	饼图	1小时（相对）	CC攻击来源的接入DDoS高防线路分布，如电信、联通和BGP等。	-
被攻击网站Top10	环图	1小时（相对）	被攻击最多的10个网站。	-

- **访问中心**

  访问中心展示被DDoS保护的网站目前的总体被访问状况，包括PV/UV趋势与带宽峰值、访问者分布、流量线路分布、客户端类型分布、请求分布、被访问网站分布等。

图表	类型	默认时间范围	描述	样例
PV	单值	1小时（相对）	请求总数。	100000
UV	单值	1小时（相对）	独立的访问客户端总数。	100000
流入流量	单值	1小时（相对）	网站的流入流量总和，单位为MB。	300 MB
网络in带宽峰值	单值	1小时（相对）	网站请求的流入流量速率的最高	100 Bytes/s

图表	类型	默认时间范围	描述	样例
			峰值，单位为Bytes/s。	
网络out带宽峰值	单值	1小时（相对）	网站请求的流出流量速率的最高峰值，单位为Bytes/s。	100 Bytes/s
流量带宽趋势	双线图	1小时（整点）	每分钟的网站流入流出流量的趋势图（单位KB/S)	-
请求与拦截趋势	双线图	1小时（整点）	每分钟的请求和拦截的CC攻击请求总数的趋势图。单位为个/分钟。	-
PV/UV访问趋势	双线图	1小时（整点）	每分钟的PV与UV的趋势图。单位为个。	-
访问者分布	世界地图	1小时（相对）	访问者PV在来源国家的分布。	-
访问者热力图	高德地图	1小时（相对）	访问者在地理位置上的访问热力图。	-
流入流量分布	世界地图	1小时（相对）	流入流量总和在来源国家的分布。单位为MB。	-
流入流量分布	中国地图	1小时（相对）	流入流量总和在来源省份的分布。单位为MB。	-
接入线路分布	环图	1小时（相对）	访问者来源的接入DDoS高防线路分布，如电信、联通和BGP等。	-
流入流量网络提供商分布	环图	1小时（相对）	访问者通过网络运营商接入的流入流量分布。如	-

图表	类型	默认时间范围	描述	样例
			电信、联通、移动、教育网等。单位为MB。	
访问最多的客户端	表格	1小时（相对）	前100个访问最多的客户端信息，包括IP、地域城市、网络、请求方法分布、流入流量、错误访问次数、拦截的CC攻击次数等。	-
访问域名	环图	1小时（相对）	前20个被访问最多的域名。	-
Referer	表格	1小时（相对）	前100个最多的跳转Referer URL、主机以及次数等。	-
客户端类型分布	环图	1小时（相对）	前20个被访问最多的User Agent（用户代理），如iPhone、iPad、Widnows IE、Chrome等。	-
请求内容类型分布	环图	1小时（相对）	前20个最多的请求内容类型，如HTML、Form、JSON、流数据等。	-

5.4.5 费用说明

DDoS高防日志功能按照日志服务的收费项进行计费，未产生日志数据则不计费。日志服务为按量计费模式，并为您的DDoS专属Logstore提供专属的免费额度。

DDoS高防日志功能提供日志的采集、存储、实时查询分析、报表等功能点，依赖于日志服务的日志数据的实时查询与分析功能。该功能的收费取决于日志服务的计费模式。日志服务计费方式为按量计费，并为您的DDoS专属Logstore提供专属的免费额度，具体费用与您的日志量有关。如果您开通了日志服务，但没有为任何网站开启日志功能，则不收费。



说明：

为了防止网站被恶意攻击时难以溯源、排查或报警取证，建议您设置日志保存周期为30天以上。

扣费与欠费

日志服务实行后收费的模式，扣费周期为天。关于扣费与欠费的详细说明请参考扣费与欠费。

计费项

计费项	说明
读写流量	• 读写流量根据传输的流量计算，传输流量为压缩后的大小。DDoS日志一般有5~10的倍压缩率。 • 读写流量也包括通流式消费接口产生读取流量，一般通过API/SDK、消费组SDK等进行操作。根据压缩后的传输流量计算，通过API/SDK可以开启压缩。 **说明：** 在日志服务的控制台中，日志消费下的预览功能也会产生微量流式消费流量。 • 对数据进行基于索引的查询、分析产生的流量免收读写流量费。例如通过控制台上进行日志查询分析、报表和告警等流量不收费。
存储空间	存储空间为压缩后原始数据量与索引数据量之和。
索引流量	• 索引流量根据实际索引字段进行计算。在写入时一次性收取费用。DDoS的日志默认开启全索引。 • 对同时建全文索引（FullText)、键值索引（KeyValue）字段，只计算一次流量。 • 索引建立后占用存储空间，所以会产生存储空间费用。

计费项	说明
活跃Shard租用	Shard 租用只统计当前读写 Shard 的数量。已经被合并或分裂的 Shard 不收取租用费。  说明： 日志服务默认为您创建2个Shard，并开启自动分裂Shard功能。一般每个Shard可以支持430 GB/天的数据写入量。
读写次数	日志写入日志服务的次数，由您的日志产生速度决定。后台实现机制会尽量减少读写次数。
外网读取流量	日志服务收集上来的日志数据被被外网程序读取消费所产生的数据流量。

免费额度

在以下情况，日志服务不收费：

- 开通了日志服务产品，但尚未为任何网站开启DDoS高防日志采集功能。
- 开启DDoS日志功能的网站日志量在免费额度之内。
- 基于索引的查询分析、报表和告警免收读写流量费。

日志服务为您的DDoS专属Logstore提供专属的免费额度，若您的数据量小于免费额度，则不收费。

计费项	免费额度
读写流量	• 读取流量：正常计费 • 写入流量：免费额度 30GB/天
存储空间	3天
索引流量	100 GB/天
活跃Shard租用	4 个*天/月
读写次数	• 读取次数：正常计费 • 写入次数：免费额度 1 百万次/天
外网读取流量	0

 说明：

您的日志数据保存时间默认设置为3天，当您修改为3天以上时，超出的部分收费。

计费方式

当开启日志分析的网站总日志量超过免费额度时，日志服务将对超出部分按照日志量收取费用。

计费项	超出部分价格（公共云）	超出部分价格（金融云）
读写流量（元/GB）	0.18	0.342
存储空间（元/GB/天）	0.0115	0.01725
索引流量（元/GB）	0.35	0.665
活跃Shard租用（元/天）	0.04	0.04
读写次数（元/百万次）	0.12	0.12
外网读取流量（元/GB）	0.8	0.8

计费样例

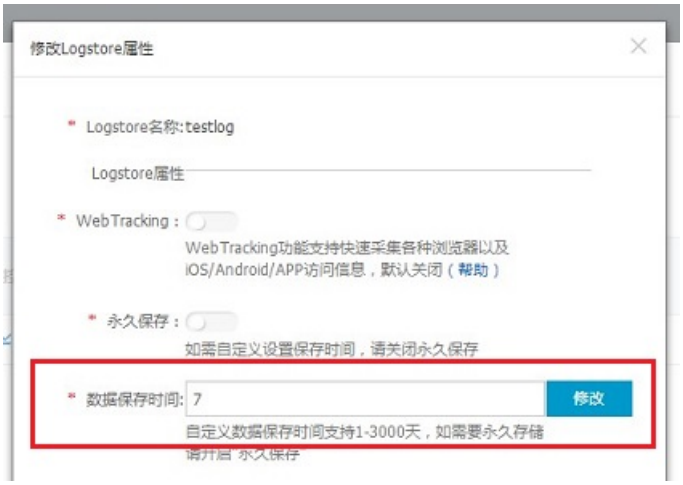
- **免费额度**：平均每条日志1600字节左右，每天6千万条日志，存储周期3天，则总日志量约为 96 GB/天，符合免费额度。
- **索引**：日志量为 150 GB/天，则以 50 GB（150GB - 100GB）计费，为 0.35 x 50 = 4.375 元/天。
- **写入传输**：日志量为 300GB/天，按照6压缩比计算，实际压缩大小约为 50GB，则以 20GB（50GB - 30GB）计费，为 0.18 x 20 = 3.6 元/天。
- **存储空间大小**：
  - 每天 10GB 数据，压缩后为 2GB，索引流量10 GB。存储周期为 30 天，则30 天后累计 最大存储量为 30 ×（10+2）= 360GB，去掉3天免费额度后 为 27 *（10+2）= 324 GB，一天存储最大收费为 0.0115 × 324 = 3.726 元。
  - 每天 1GB 数据，压缩后为 200MB，全文和字段索引（大小为 1GB）。存储周期为 30 天，则30 天后累计最大存储量为 30 ×（1000+200）≈ 36GB，去掉3天免费额度后为 27 *（1000+200）≈ 32.4 GB，一天存储最大收费为 0.0115 × 32.4 = 0.3726 元。
- **活跃Shard租用**：目前有 10 个 Shard，7 个 Shard 的状态为读写，另外 3 个 Shard 已经被合并为只读，DDoS专属的Logstore总体每天只收取 3个（7个- 4个）Shard的租赁费 0.12 元/天。
- **读写次数**：每天网站日志为100亿条，写入次数约为50W次（平均2000条/次），免费。
- **外网流量**：2GB日志服务数据被投递至非阿里云产品，产生外网读取流量1.6元。

计费FAQ

- 如何修改网站日志的存储时间？

  1. 登录日志服务控制台，单击Project名称，进入Logstore列表。DDoS日志的默认Project为`ddos-pro-project-阿里云账户ID-cn-hangzhou`（大陆地区DDoS高防实例）和`ddos-pro-project-阿里云账户ID-ap-southeast-1`（其他地区DDoS高防实例）。
  2. 单击操作列的修改。
  3. 在弹出页面中修改数据保存时间，并单击修改。

  图 **5-42:** 修改日志保存时间



- 如何查看当前日志量并预估费用？
  - 在阿里云的费用管理中心中查看每天定时刷新的费用计量数据。
  - 1. 登录DDoS高防IP控制台，单击左侧菜单中的全量日志。
    2. 选择需要查看日志量的网站，单击右侧的日志分析。
    3. 在查询框中输入以下查询语句，时间范围为昨天（整点时间）：

    ```
    __topic__: ddos_access_log | select count(1) as PV
    ```

    4. 单击查询，选择统计图表页签，图表类型为表格。

图 5-43: 查看日志量



可以得到昨天一天的数据量，结合您当前配置的日志保存时间预估费用。

- 如何配置日志服务在产生大量日志时告警？

  当采集到大量DDoS高防日志时，可能会超出日志服务的免费额度，产生一定的费用。如果您希望在有此类风险时收到告警通知，可以配置日志服务在产生大量日志时发送告警。

  1. 登录DDoS高防IP控制台，单击左侧菜单中的**全量日志**。
  2. 选择需要查看日志量的网站，单击右侧的**日志分析**。
  3. 在查询框中输入以下查询语句，单击**查询**：

     ```
     *| select count(1) as PV
     ```

  4. 单击查询页面右上角的**另存为快速查询**，填写快速查询的相关信息，例如`ddos-metering-pv`，单击**确定**。
  5. 单击**另存为告警**，并参考下图创建一个告警配置。规定每5分钟检查过去1小时的日志量，如果大于560万条则发送告警。

图 **5-44:** 告警规则



说明：

保证每天日志量小于100GB免费额度的情况下，推算每小时平均导入量为：100GB ÷ 1600 字节/条 ÷ 24小时 ≈ 280 万条。示例为2倍每小时日志量，即560万条的情况下报警，可以根据实际情况和需要做适当调节。

5.5 ActionTrail访问日志

5.5.1 简介

目前，阿里云操作审计已经与日志服务打通，提供日志实时采集与分析等功能。操作审计收集到的操作日志数据会实时投递到日志服务中，日志服务针对这部分日志可以提供实时查询与分析、仪表盘方式展示等丰富功能。

伴随着越来越多的企业采用信息化、云计算技术来提高效率与服务质量。针对企业组织的网络、设备、数据的攻击从来没有停止过升级，这些针对性攻击一般以牟利而并是不破坏为目的，且越来越善于隐藏自己，因此发现并识别针这些攻击也变得越来越有挑战。

作为审计与安全回溯的基础，企业IT与数据资源的操作的日志一直以来是重中之重。随着网络信息化的成熟发展，并伴随《网络安全法》的深入落实，企业组织也越来越重视操作日志的保存与分析，其中云计算中的资源的操作记录是一类非常重要的日志。

阿里云操作审计（ActionTrail）云产品会记录您的云账户资源操作，提供操作记录查询，并可以将记录文件保存到您指定的OSS或日志服务中。利用 ActionTrail保存的所有操作记录，您可以实现安全分析、资源变更追踪以及合规性审计。

ActionTrail收集云服务的API调用记录，包括用户通过控制台触发的API调用记录，规格化处理后将操作记录以JSON形式保存并支持投递。一般情况下，当用户通过控制台或SDK发起操作调用之后，ActionTrail会在十分钟内收集到操作行为。

目前，阿里云操作审计已经与日志服务打通，提供日志实时采集与分析等功能。操作审计收集到的操作日志数据会实时投递到日志服务中，日志服务针对这部分日志可以提供实时查询与分析、仪表盘方式展示等丰富功能。

功能优势

- **配置简单**：轻松配置即可实时采集日志。操作步骤及日志字段请参考操作步骤。
- **实时分析**：依托日志服务，提供实时日志分析，并提供开箱即用的报表中心，对重要云资产的操作了如指掌，并可实时挖掘细节。
- **实时告警**：支持基于特定操作定制准实时的监测与告警，确保关键业务异常时可及时响应。
- **生态体系**：支持对接其他生态如流计算、云存储、可视化方案，进一步挖掘数据价值。
- **免费额度**：提供每月500MB免费导入与存储额度，并可自由扩展存储时间，以便合规、溯源、备案等。支持不限时间的存储，存储成本低至0.35元/GB/月。计费说明请查看计费方式。

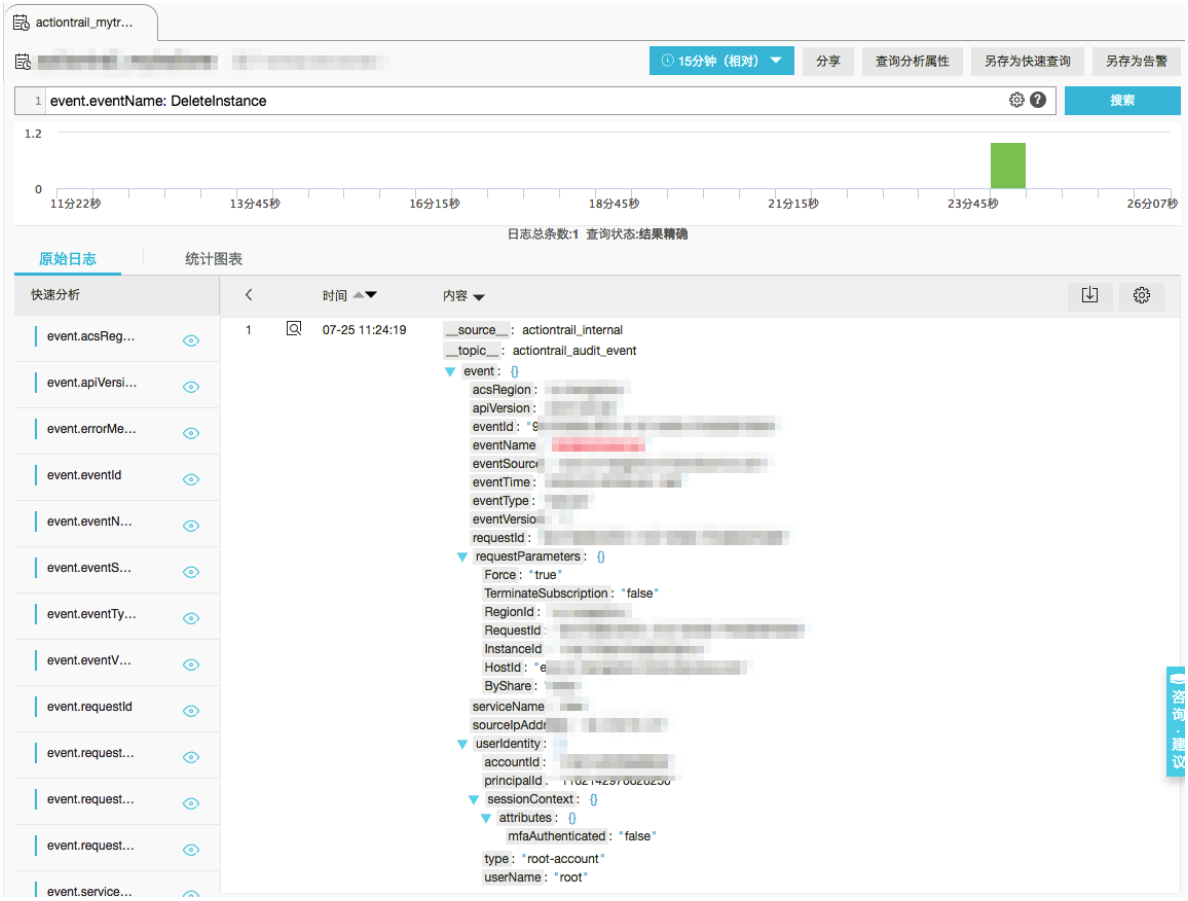
应用场景

- **操作异常排查与问题分析**

实时监控所有名下的云资源操作，支持针对操作异常进行实时排查与问题分析。意外删除、高危操作等均可通过日志记录溯源。

例如，查看ECS释放操作日志：

图 5-45: 查看ECS释放操作日志

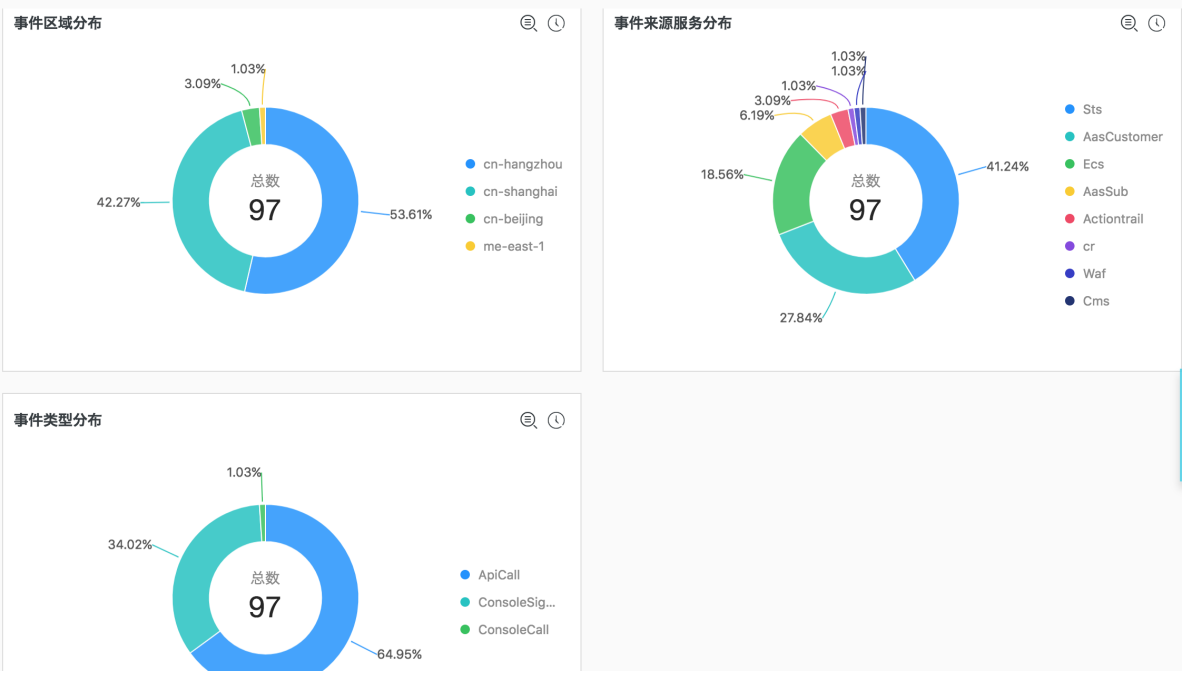


- **重要资源操作的分布与来源追踪**

  您可以通过分析日志内容，对重要资源操作的分布和来源进行追踪与溯源，并依据分析结果指定并优化对应策略。

  例如，查看删除RDS的操作者的国家分布：

图 **5-46: RDS删除操作**

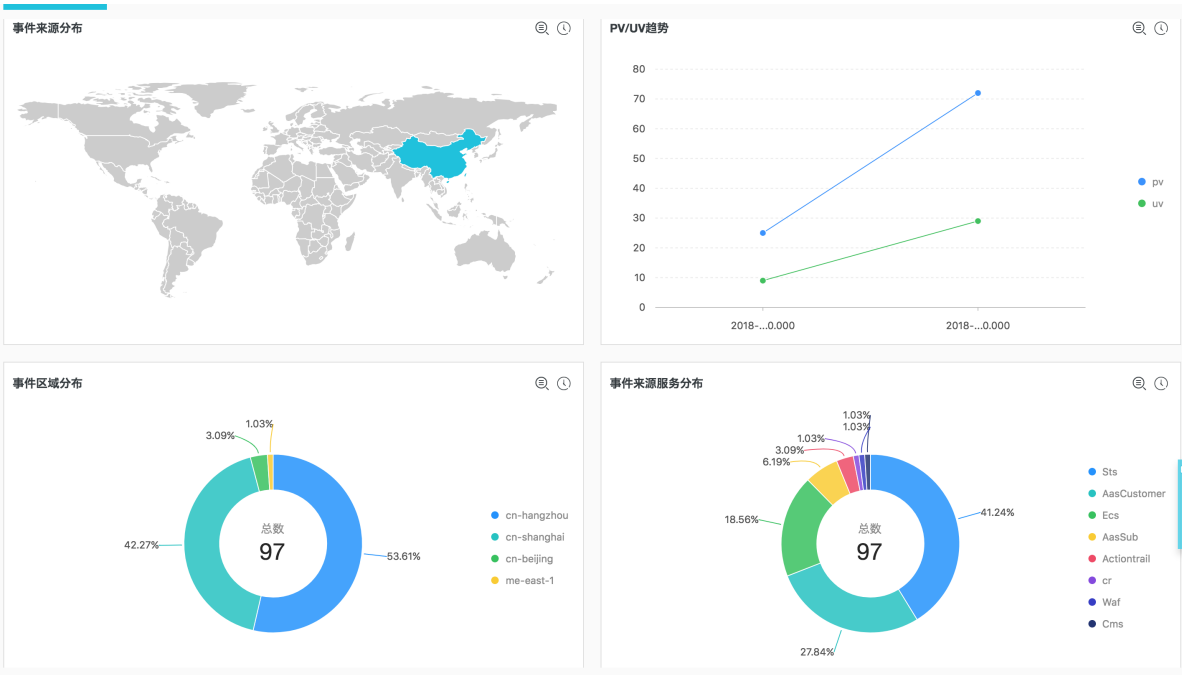


- 查看资源操作分布

  您可以通过SQL查询语句，对采集到的ActionTrail操作日志进行实时查询与分析，查看所有资源操作的分布、时间趋势等其他运维动作，辅助运维人员对资源运行状况进行实时监控，运维可靠性指标一目了然。

  例如，查看失败的操作的趋势：

图 **5-47:** 失败操作的趋势

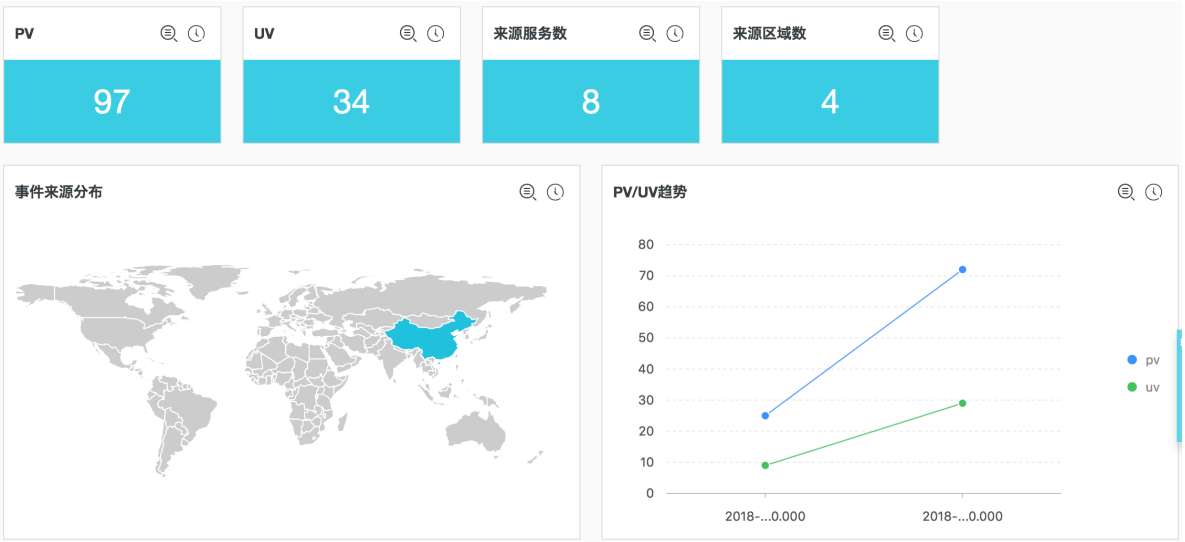


- 实时运营数据分析

  根据运营需求定制多样化的查询语句、针对不同数据需求定制快速查询、分析仪表盘等，还可以对资源使用状况、用户登录情况等数据定制实时数据大盘。

  例如，查看来自各个网络运营商的操作者的频率分布：

图 **5-48:** 来自各个网络运营商操作者的频率分布



5.5.2 操作步骤

目前，阿里云操作审计已经与日志服务打通，操作审计收集到的操作日志数据会实时投递到日志服务中。本文档为您介绍操作审计日志的日志字段及采集步骤。

前提条件

1. 开通日志服务。
2. 开通操作审计服务。

配置步骤

1. 登录ActionTrail控制台。
2. 单击左侧导航栏的**跟踪列表**，进入**跟踪列表**界面。
3. 单击右上角的**创建跟踪**，进入**创建跟踪**配置页面。
4. 填写跟踪配置。
   1. 填写**跟踪名称**。
   2. 将审计事件投递到OSS Bucket（可选）。

      详细信息请参考创建跟踪。
   3. 选择日志服务**Project**区域。
   4. 填写日志服务**Project**名称。

      此处填写的Project为存储操作审计日志的Project。您可以填写已选择的地域下的Project，或者输入一个新的Project名称，将日志投递到新的Project中。
   5. 开启**日志记录**。

      单击开启**日志记录**功能。开启功能后，您的ActionTrail记录的云资源操作日志会实时投递到日志服务。

图 5-49: 填写跟踪配置



5. 单击提交，结束配置。

至此，您已成功新建了跟踪任务，在跟踪列表可以查看已创建的跟踪信息。

> 说明：
> 如果您是第一次配置ActionTrail日志采集，请根据页面提示进行授权。授权后ActionTrail有权限将ActionTrail日志分发到您的Logstore中。请在授权完成后再次单击提交，结束配置。

图 5-50: 跟踪列表



限制与说明

- 一个账户只能创建一个跟踪。

  跟踪是您将审计事件投递到OSS Bucket或日志服务Logstore中的任务，当前一个账户在所有区域内仅支持配置一个跟踪，这个跟踪中可以将所有区域内的审计事件同时投递到OSS和Logstore，或二者之一。

- 如果已创建追踪，只能在创建时所在区域对这个追踪进行操作。

  如果您已经创建了一个追踪，后续只能在创建追踪时所在的区域下查看跟踪、修改跟踪或删除跟踪。例如您需要配置日志服务跟踪，但已创建了一个OSS跟踪，请修改已创建的OSS追踪，在该追踪中添加日志服务配置即可。

- 专属日志库不支持写入其他数据。

  专属日志库用于存入ActionTrail的云产品操作日志，因此不支持写入其他数据。其他查询、统计、报警、流式消费等功能没有限制。

- 按量计费。

  ActionTrail日志采集功能按照日志服务的收费项进行计费。日志服务为按量计费模式，并提供一定的免费额度。详细计费说明请参考计费方式。

查询分析

完成跟踪配置后，您可以在跟踪列表界面单击日志服务列下的日志分析和日志报表，对采集到的日志数据进行实时查询分析。

- 日志分析：进入日志查询分析界面。

日志服务提供查询与分析功能，您可以在该页面对采集到的ActionTrail日志数据进行实时查询与分析。

日志服务定义了一系列查询语法和分析语法，支持多种复杂场景下的日志查询。查询与分析语法请参考查询语法和分析语法。

另外，您还可以在查询界面将当前查询条件另存为快速查询和告警，定时监控重要日志数据，在异常情况时发送告警通知。详细步骤请参考设置告警。

- 日志报表：进入仪表盘界面。

  日志服务为您内置了ActionTrail专属仪表盘，为您全方位展示事件类型和事件来源分布等实时动态。

  您也可以修改专属仪表盘、创建自定义仪表盘，为您的仪表盘添加多种场景下的自定义分析图表。关于仪表盘的更多信息，请参考仪表盘。

默认配置

完成配置后，日志服务会为您创建专属Project和专属Logstore，ActionTrail的云资源操作日志会实时投递至该Logstore中。另外，日志服务同时为您创建了1个仪表盘，您可以实时查看云资源操作动态。专属日志库和专属Logstore等默认配置请参见下表。

表 **5-10:** 默认配置

默认配置项	配置内容
Project	您在创建跟踪时选择或自定义的Project。
Logstore	默认为您创建Logstore。Logstore名称为`actiontrail_`跟踪名称。 ActionTrail的所有日志都会保存在这个Logstore中。
地域	您在创建跟踪时选择的地域。
Shard	默认为您创建2个Shard，并开启自动分裂*Shard*功能。
日志存储时间	默认永久保存。 若您需要修改日志存储时间，可以自定义修改为1~3000天。详细步骤请参考操作*Logstore*。
仪表盘	默认为您创建一个仪表盘： • 中文环境：`actiontrail_`跟踪名称`_audit_center_cn` • 英文环境：`actiontrail_`跟踪名称`_audit_center_en`

日志字段

字段名	名称	例子
__topic__	日志主题。	固定为`actiontrail_audit_event`
event	事件主体，JSON格式。事件主体的内容随事件变化。	*event*示例
event.eventId	事件的ID，唯一标示这次事件的ID。	07F1234-3E1D-4BFF-AC6C-12345678
event.eventName	事件名称。	CreateVSwitch
event.eventSource	事件来源。	http://account.aliyun.com:443/login/login_aliyun.htm
event.eventType	事件类型。	ApiCall
event.eventVersion	ActionTrail的数据格式版本，目前固定为1。	1
event.acsRegion	事件所在的区域。	cn-hangzhou
event.requestId	操作云服务的请求ID。	07F1234-3E1D-4BFF-AC6C-12345678
event.apiVersion	相关API的版本。	2017-12-04
event.errorMessage	事件失败的错误信息。	unknown confidential
event.serviceName	事件相关服务名称。	Ecs
event.sourceIpAddress	事件相关的源IP。	1.2.3.4
event.userAgent	事件相关的客户端Agent。	Mozilla/5.0 (...)
event.requestParameters.HostId	请求相关参数中的主机ID。	ecs.cn-hangzhou.aliyuncs.com
event.requestParameters.Name	请求相关参数中的名称。	ecs-test
event.requestParameters.Region	请求相关参数中的域。	cn-hangzhou
event.userIdentity.accessKeyId	请求所使用的AccessKey ID。	25**********
event.userIdentity.accountId	请求相关账户的ID。	123456
event.userIdentity.principalId	请求相关账户的凭证ID。	123456

字段名	名称	例子
event.userIdentity.type	请求相关账户的类型。	root-account
event.userIdentity.userName	请求相关账户的类型。	root

event示例

```
{
  "acsRegion": "cn-hangzhou",
  "additionalEventData": {
    "isMFAChecked": "false",
    "loginAccount": "test1234@aliyun.com"
  },
  "eventId": "7be1e173-1234-44a1-b135-1234",
  "eventName": "ConsoleSignin",
  "eventSource": "http://account.aliyun.com:443/login/login_aliyun.htm
",
  "eventTime": "2018-07-12T06:14:50Z",
  "eventType": "ConsoleSignin",
  "eventVersion": "1",
  "requestId": "7be1e173-1234-44a1-b135-1234",
  "serviceName": "AasCustomer",
  "sourceIpAddress": "42.120.75.137",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.
36",
  "userIdentity": {
    "accessKeyId": "25****************",
    "accountId": "1234",
    "principalId": "1234",
    "type": "root-account",
    "userName": "root"
  }
}
```

5.6 NAS访问日志

5.6.1 简介

日志服务支持对NAS访问日志进行查询分析，并提供开箱即用的报表与实时监控的报警功能。

阿里云文件存储（Network Attached Storage，简称 NAS）是面向阿里云 ECS 实例、E-HPC 和容器服务等计算节点的文件存储服务，提供标准的文件访问协议，用户无需对现有应用做任何修改，即可使用具备无限容量及性能扩展、单一命名空间、多共享、高可靠和高可用等特性的分布式文件系统。

到达NAS的每一条请求都会产生一条访问日志。该日志记录了这次访问请求的详细信息，如用户的操作类型，目标对象，响应状态等。

日志服务支持对NAS访问日志进行查询分析，并提供开箱即用的报表与实时监控的报警功能。您可以通过对NAS访问日志的自定义查询分析，统计特定场景下的访问事件、审计敏感操作、诊断问题，通过默认仪表盘管理数据看板、实时查看访问数据，并通过报警功能实时监控访问动态、配置多种场景下的告警任务等。

功能优势

- **配置简单**：轻松配置即可实时采集日志。操作步骤及日志字段请参考操作步骤。
- **实时分析**：依托日志服务，提供实时日志分析，并提供开箱即用的报表中心，对重要云资产的操作了如指掌，并可实时挖掘细节。
- **实时告警**：支持基于特定操作定制准实时的监测与告警，确保关键业务异常时可及时响应。
- **生态体系**：支持对接其他生态如流计算、云存储、可视化方案，进一步挖掘数据价值。
- **默认配置免费**：支持自由扩展存储时间，以便合规、溯源、备案等。日志数据默认保存7天，同时支持设置为永久保存，超出7天的部分存储成本低至0.35元/GB/月。计费说明请查看计费方式。

限制与说明

- **专属日志库不支持写入其他数据**。

  专属日志库用于存入NAS访问日志，因此**不支持写入其他数据**。其他查询、统计、报警、流式消费等功能没有限制。

- **仅支持NFS协议的文件系统**。

  日志分析目前仅支持NFS协议的文件系统，后续会陆续支持其他协议类型。

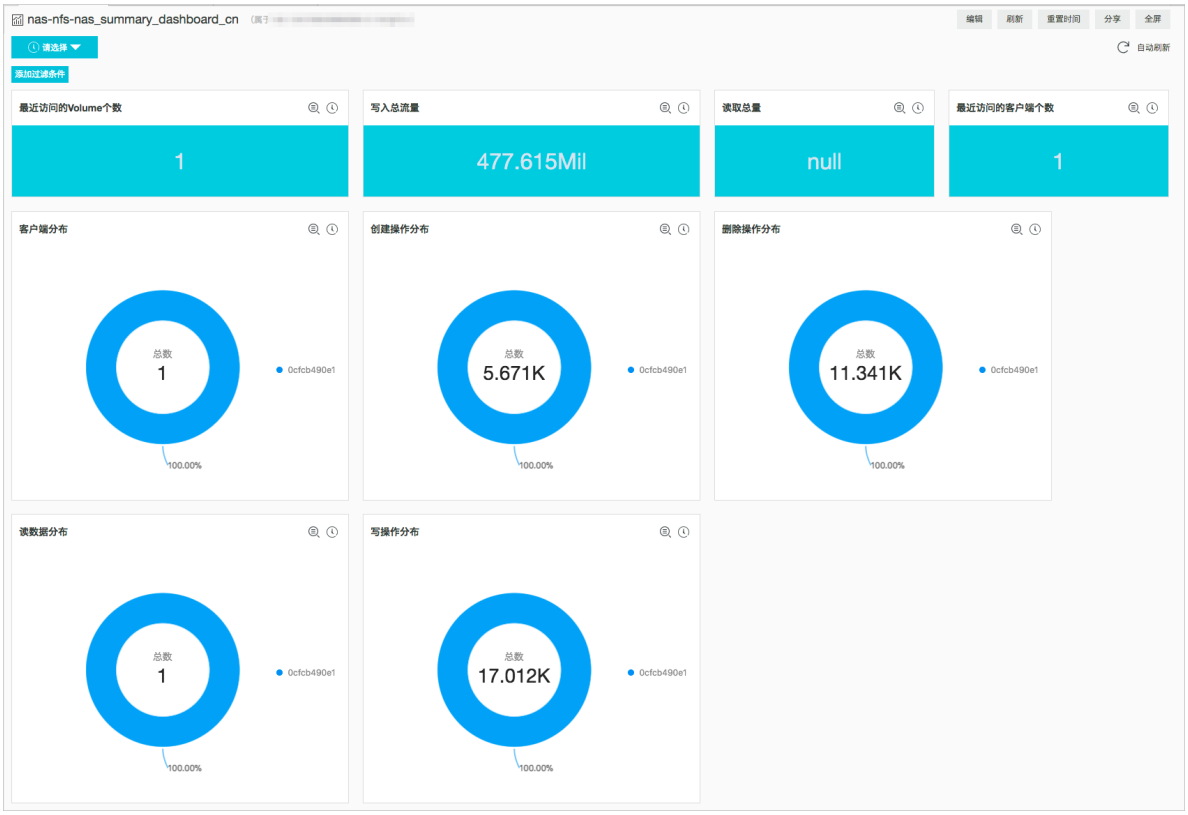
- **按量计费**。

  NAS日志采集功能处于测试阶段，默认情况下不收费。您的NAS日志在日志服务中默认保存时间为7天，如果您调整为更长时间，则需要为超出的使用量收费。日志服务为**按量计费**模式，详细计费说明请参考计费方式。

应用场景

- **浏览NAS各个Volume的读写操作情况**

  在总览视图中，可以看到活跃的Volume个数、总的读写流量，以及每个Volume的创建、删除、读、写操作的次数、分布情况。



- 查看**NAS**访问明细

  在访问明细中，可以看到某些操作在时间上的分布，以及各种成功或失败的操作明细、失败的原因，例如：

  — 数据读写流动趋势，从哪些IP写到哪些Volume，再由哪些IP读取。

  — 各个Volume访问的文件个数和比例。

  — 各个Volume访问QPS。

  — 各个Volume每分钟读写总流量、平均流量。

  — Top读写客户端IP。

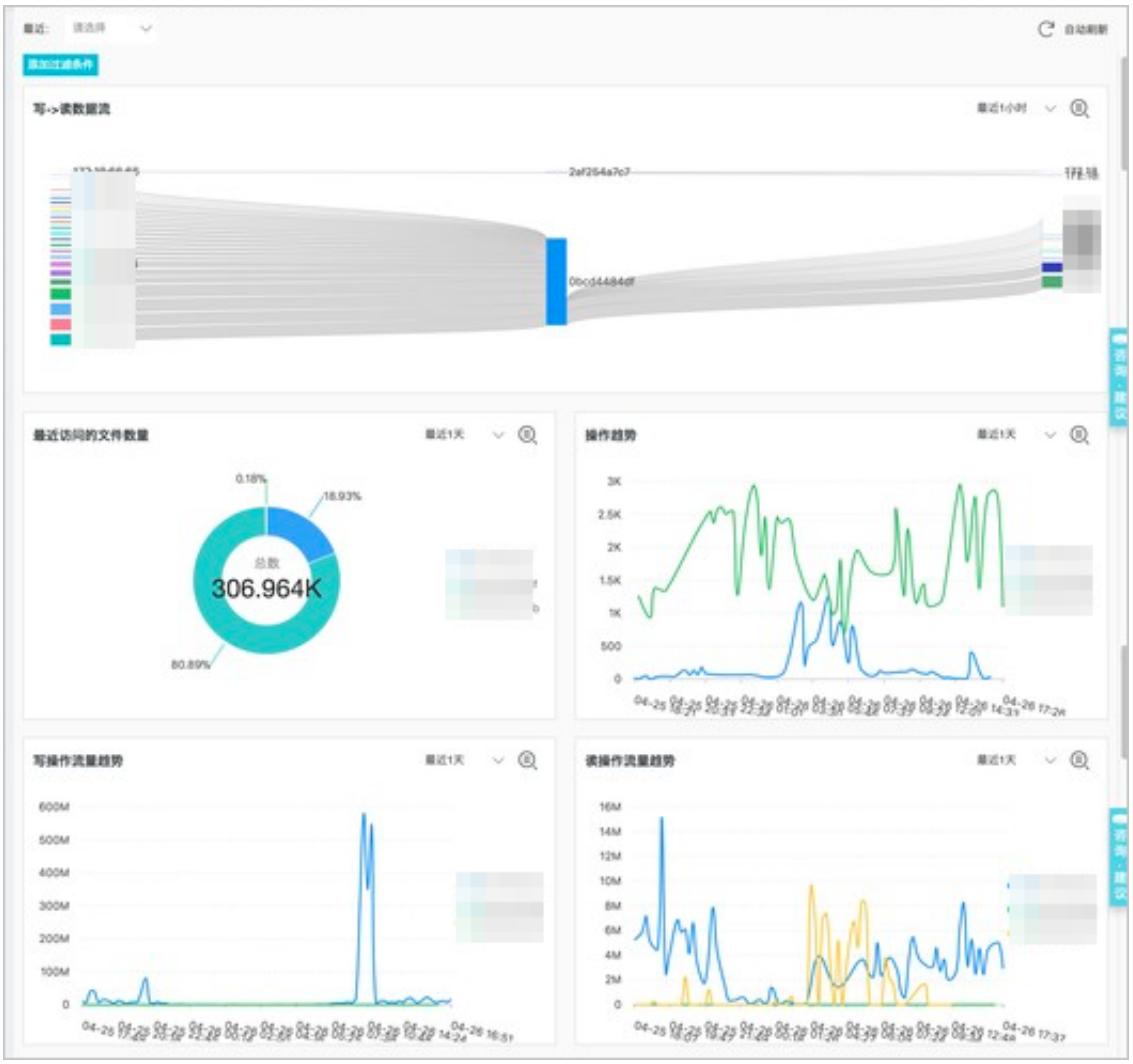
  — 操作错误的Top IP，用于诊断操作错误来源。

  — 热点的读、写、创建、删除操作次数。

  — 热点访问文件的Inode。

  — 各种异常操作的次数，如鉴权失败，操作失败等。

  — 通过AuthRc和NFSProtocolRc判断各种操作状态的分布。

- 

- 敏感操作审计

  审计视图可以用于查看您对NAS的一些敏感操作，比如创建、删除目录或文件，以及读写最多的文件。



- 搜索访问日志
  - 执行以下查询语句，搜索鉴权失败的日志，用于识别一些权限异常的操作。

    ```
    AuthRc > 0
    ```

  - 执行以下查询语句，搜索操作失败的日志，查看哪些操作结果不正确。

    ```
    NFSProtocolRc > 0
    ```

5.6.2 操作步骤

NAS访问日志可以实时投递到日志服务，该日志记录了访问事件的详细信息，如用户的操作类型，目标对象，响应状态等。本文档为您介绍采集NAS日志的日志字段和采集步骤。

前提条件

1. 开通日志服务和NAS产品。
2. 创建文件系统，添加挂载点，并挂载文件系统。

   详细步骤请参考创建文件系统、添加挂载点和挂载文件系统。

配置步骤

1. 主账号登录NAS控制台。
2. 根据控制台上方的提示信息，单击**申请**申请开通资格。

   NAS现已开通用户级监控，可以实时监控NAS的多项I/O指标，您需要在控制台上方申请开通资格才有权限配置。
3. 单击左侧导航栏的**日志分析 > 日志管理**，进入**日志管理**页面。
4. 选择**地域**。
5. RAM授权。

   如果您是第一次配置NAS日志采集，请根据控制台上方的提示信息，单击**授权入口**进行授权。授权后NAS将有权限将NAS访问日志分发到您的Logstore中。
6. 返回到**日志管理**页面。
7. 单击页面右上角的**新建日志转储**。
8. 选择**地域**和**文件系统ID/名称**，单击**确认**。

成功执行以上操作后，日志服务将为您创建一个Project和Logstore，您的NAS操作日志将实时投递到日志服务的默认Logstore中，您可以在**日志管理**页面查看文件系统对应的日志服务Project和Logstore。此外，您还可以：

- 单击点击前往，跳转到日志服务控制台查看日志。
- 单击停止，停止将日志转储到日志服务中。

更多操作

在日志管理页面单击点击前往，进入日志服务控制台的查询分析页面。您可以：

- 查询分析日志

  日志服务定义了一系列查询语法和分析语法，支持多种复杂场景下的日志查询。查询与分析语法请参考查询语法和分析语法。
- 查看仪表盘

  日志服务为您内置了3个NAS专属仪表盘，分别为为您全方位展示事件类型和事件来源分布等实时动态。

  您也可以修改专属仪表盘、创建自定义仪表盘，为您的仪表盘添加多种场景下的自定义分析图表。关于仪表盘的更多信息，请参考仪表盘。
- 配置告警

  另外，您还可以在查询界面将当前查询条件另存为快速查询和告警，定时监控重要日志数据，在异常情况时发送告警通知。详细步骤请参考设置告警。

默认配置

表 5-11: 默认配置

默认配置项	配置内容
Project	默认为您创建1个Project • 中文环境：`nas-AliUid-cn-RegionId` • 英文环境：`nas-AliUid-en-RegionId`
Logstore	默认为您创建1个Logstore`nas-nfs`。NAS日志采集功能产生的所有日志都会保存在这个Logstore中。 **说明：** 日志分析目前仅支持NFS协议的文件系统，后续会陆续支持其他协议类型。
地域	默认保存在当前文件系统所在的地域中。
Shard	默认为您创建2个Shard，并开启自动分裂*Shard* 功能。

默认配置项	配置内容
日志存储时间	默认保存7天，7天以上的日志自动删除。 若您需要更长的存储时间，可以自定义修改。如果修改为更长的存储时间，则需要为超出的部分付费。收费模式请查看计费方式。
仪表盘	默认为您创建三个仪表盘，分别为： • `nas-nfs-nas_audit_dashboard` • `nas-nfs-nas_detail_dashboard` • `nas-nfs-nas_summary_dashboard`

日志字段

字段名	说明	示例
ArgIno	文件系统inode号。	226
AuthRc	授权返回码。	0
NFSProtocolRc	NFS协议返回码。	0
OpList	NFSv4 Procedures编号。	null
Proc	NFSv3 Procedures编号。	1
RWSize	读写大小，单位为字节。	-1
RequestId	请求ID。	5ACF5CD506EAC7A508F056DF
ResIno	lookup的资源inode号。	null
SourceIp	客户端IP。	127.0.0.1
User	用户ID。	123456789
Vers	NFS协议版本号。	3
Vip	服务端IP。	172.18.158.178
Volume	文件系统ID。	2d2794a330
microtime	请求发生时间，单位为微秒。	1523539157201995

5.7 态势感知日志

阿里云态势感知日志分析（Log Analysis）可提供态势感知风险威胁数据的实时收集、查询与实时分析、存储和分发等一站式服务。无需您开发就能快捷完成全要素风险和威胁日志数据的采集、查询与实时分析等，帮助您提升运维、运营效率。

功能说明

- 开通态势感知日志分析功能
- 自定义日志查询与分析
- 日志报表仪表盘
- 日志类别及参数说明
- 导出日志

背景信息

阿里云日志服务与态势感知产品全方位对接，提供态势感知日志采集与分析功能，帮助您全面了解、有效处理服务器的安全隐患，实现对云上资产的集中安全管理。

- 对云上资产的主机、网络以及安全日志有存储合规需求的大型企业与机构，如金融公司、政府类机构等。
- 拥有自己的安全运营中心（SOC），需要收集安全告警等日志进行中央运营管理的企业，如大型地产、电商、金融公司、政府类机构等。
- 拥有较强技术能力，需要基于云上资产的日志进行深度分析、对告警进行自动化处理的企业，如IT、游戏、金融公司等。

功能优势

- 快速：安全与主机日志分析从分钟级提升为秒级，网络日志从多小时提升为1小时级别。
- 全面：覆盖网络、主机与安全三大类共14种日志。
- 开放：与阿里云、开源生态下流计算、大数据系统融合，并对合作伙伴开放。
- 灵活：所见即所得的分析能力，支持自定义构建业务视图。

限制与说明

- 专属日志库不支持写入其他数据。

  态势感知日志将被存放在专属日志库中，该日志库不支持通过API/SDK在内的所有方式写入其他数据。专属日志库在查询、统计、报警、流式消费等均功能上无特殊限制。

- 不支持修改专属日志库的存储周期等基本设置。
- 专属日志库不收费。

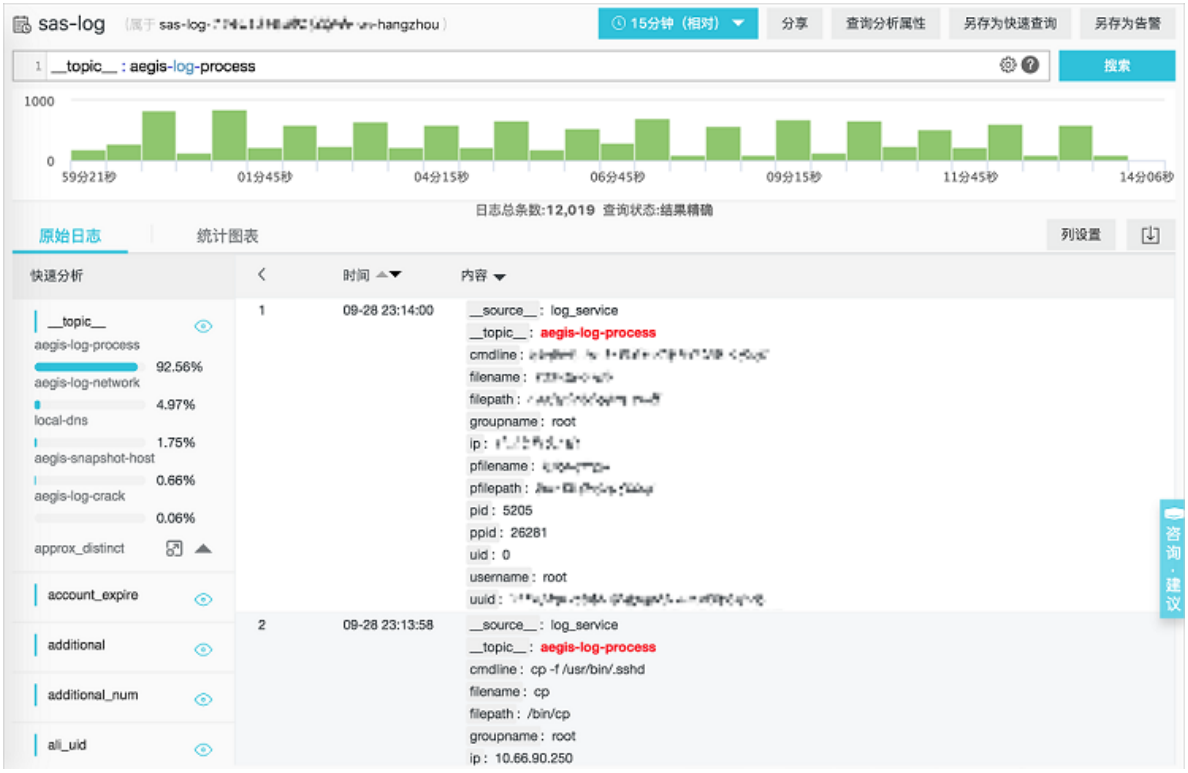
  日志服务对专属日志库不进行任何收费，日志服务产品需处于正常使用状态。

  > **说明：**
  > 日志服务欠费时，态势感知日志采集功能暂停使用，请及时补缴欠款。

应用场景

- 追踪主机与网络日志，溯源安全威胁

  根据日志中的`__topic__`字段实时检索，查看不同类型日志的时间分布，对主机日志和网络日志做大实时跟踪。



- 实时查看主机与网络活动，洞察状态与趋势

  通过**Web访问中心**等仪表盘页面实时查看主机与网络活动，及时判断当前的安全状态。



- 快速了解安全运营效率，即时反馈处理

  通过漏洞中心仪表盘查看当前安全运营效率。



5.8 DRDS SQL审计日志

5.8.1 简介

分布式关系型数据库DRDS与日志服务联合推出SQL审计与分析功能，不仅支持历史SQL记录的审计，而且提供对SQL执行状况、性能指标、安全问题的实时诊断分析能力。

您可以在 DRDS 控制台，开通SQL审计与分析功能，进行实时日志分析。同时，DRDS依托日志服务，提供开箱即用的报表中心，使您对数据库执行状况、性能、潜在安全问题了如指掌。

功能优势

- **操作简单**：轻松配置即可开启，实时进行SQL日志的审计与分析。
- **性能无损**：实时拉取DRDS节点SQL日志文件上传，对实例本身性能无影响。
- **历史问题追溯**：支持导入历史SQL日志，追溯问题。
- **实时分析**：依托日志服务，提供SQL实时分析、开箱即用的报表中心、并支持自定义报表和下钻分析，对数据库执行状况、性能、安全问题了如指掌。
- **实时告警**：支持基于特定指标，定制准实时的告警，确保关键业务异常时可及时响应。
- **价格优势**：存储费用比传统方案更低， 每GB日志存储费用低至0.58元/月，详情参考计费方式。

限制说明

- 使用SQL审计与分析功能，需要开通阿里云日志服务。
- 当前支持的区域：华东1、华东2、华北1、华北2、华南1。其它地区陆续开放中。
- 共享实例不支持SQL审计与分析功能。
- SQL审计日志默认保存30天，您也可以根据需要修改日志保存时间。
- 请勿随意删除或修改日志服务为您默认创建的Project、Logstore、索引和仪表盘设置，日志服务会不定期更新与升级SQL日志审计功能，专属日志库的索引与默认报表也会自动更新。
- 子账号使用SQL审计与分析，需要为其授予日志服务相关权限，详细说明请参考为子账号授予*DRDS SQL*审计权限。
- SQL审计功能默认为关闭状态。开启该功能后，会产生额外费用，详细收费标准请参见计费方式。如果您需要关闭该功能，请在DRDS控制台关闭。关闭后不再写入数据，可以查看历史数据，这部分历史数据会产生存储和索引费用。您可以在日志服务控制台删除Logstore以删除历史数据，删除后日志服务不再对该部分数据计费。

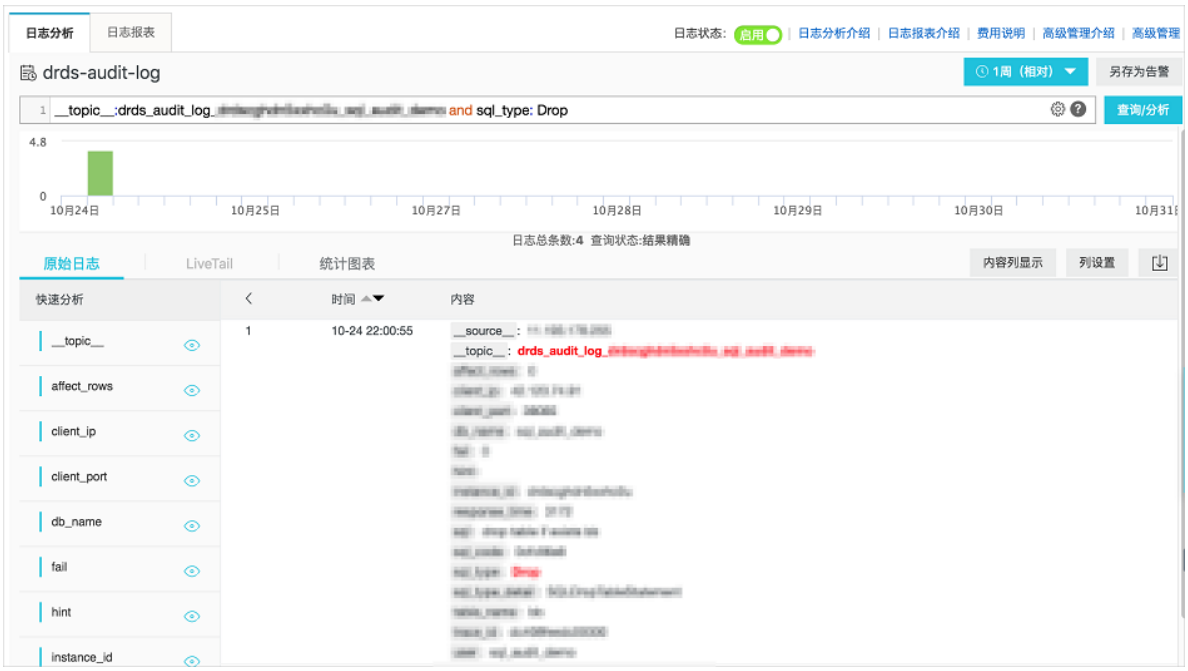
应用场景

- 排查问题SQL

开启DRDS SQL审计与分析功能后，您可以对SQL日志进行快速检索，排查定位问题。例如，需要查询是否有人执行Drop操作，可以执行如下查询：

```
sql_type: Drop
```

如下图所示，查询结果中包括SQL的执行时间、用户、执行该SQL的客户端IP地址等。



- 分析高代价SQL模板

大多数应用中，SQL通常是基于若干模板动态生成的，只是参数不同。通过日志服务的实时分析能力，您可以获取当前数据库中的高代价SQL列表。

例如执行以下查询：

```
| SELECT sql_code as "SQL模板ID", round(total_time * 1.0 /sum(
total_time) over() * 100, 2) as "总体耗时比例(%)" ,execute_times as
 "执行次数", round(avg_time) as "平均执行时间",round(avg_rows) as "平
均影响行数", CASE WHEN length(sql) > 200 THEN concat(substr(sql, 1,
200), '......') ELSE trim(lpad(sql, 200, ' ')) end as "样例SQL" FROM
 (SELECT sql_code, count(1) as execute_times, sum(response_time)
as total_time, avg(response_time) as avg_time, avg(affect_rows) as
avg_rows, arbitrary(sql) as sql FROM log GROUP BY sql_code) ORDER BY
 "总体耗时比例(%)" desc limit 10
```

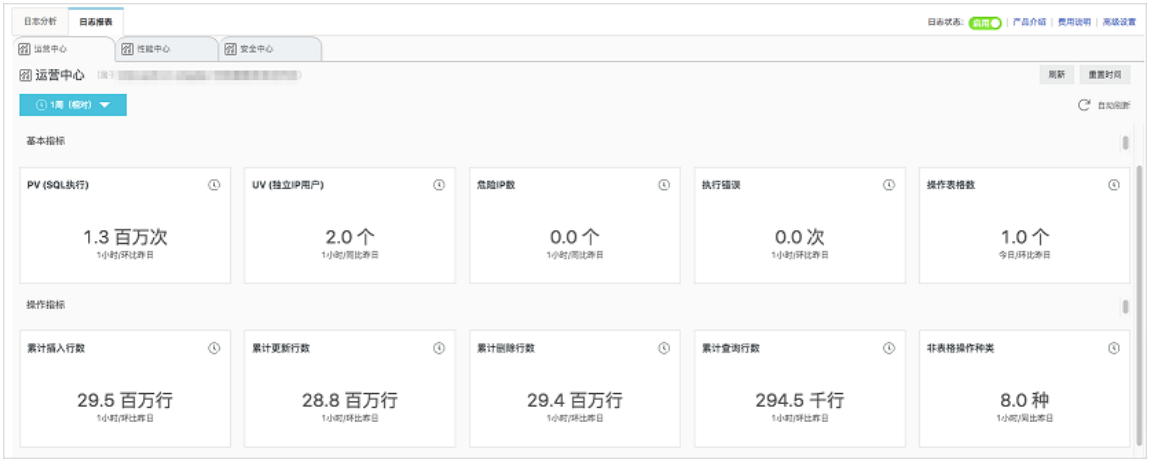
查询结果如下图所示。包括SQL模板的ID、该模板SQL占SQL的耗时比例、执行次数、平均执行时间、平均影响行数以及样例SQL。您可以基于该分析结果，找到应用中代价最高的SQL模板，对其进行优化。

SQL模板ID	总体耗时比例(%)	执行次数	平均执行时间	平均影响行数	样例SQL
ac0c0ca2	34.97	38116	10343.0	1.0	
4f3b8c1d	27.22	30212	10157.0	8.0	

- 日志统计与报表

  为了便于分析问题，DRDS SQL审计分析结合日志服务，为您提供了开箱即用的报表，让您对数据库的执行状况、性能和潜在安全问题进行实时诊断分析，报表页面如下图所示。

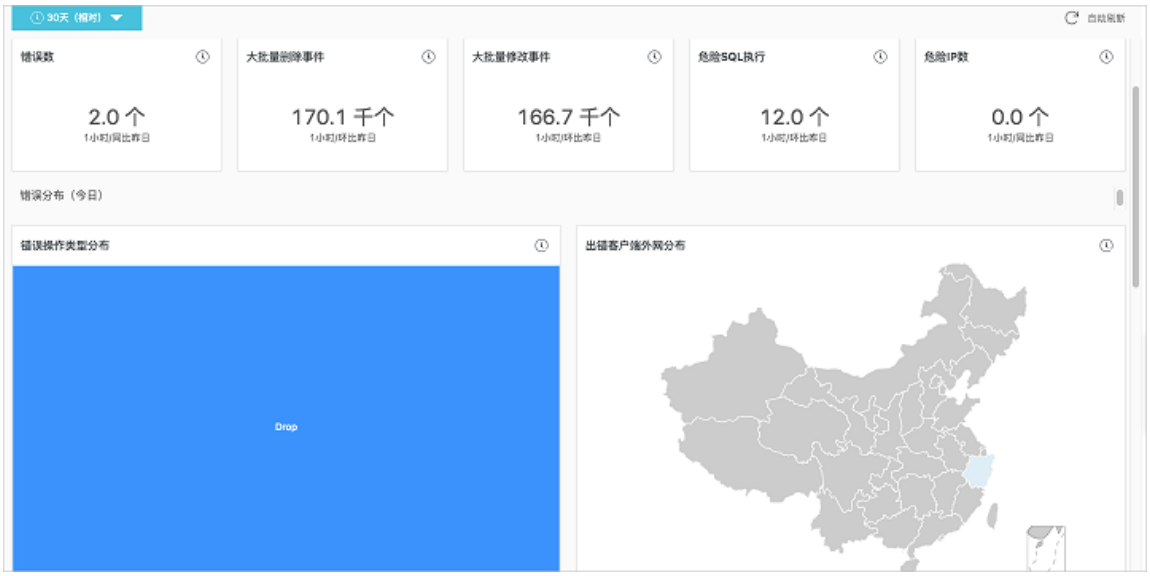
  - 运营中心



  - 性能中心



  - 安全中心



5.8.2 开启SQL日志审计

SQL日志审计功能默认关闭，您可以在DRDS控制台上手动开启。默认情况下，只对开启SQL日志审计功能之后产生的日志数据进行审计分析，您也可以导入部分历史数据。

前提条件

- 开通阿里云日志服务。
- 已购买DRDS专享实例，并创建了数据库。
- 默认情况下，只有主账号可以开启并操作SQL日志审计功能。子账号开启和操作SQL日志审计功能前，需要先授予日志服务相关权限。详细步骤请参考子账号授权。

背景信息

SQL审计功能默认为关闭状态。您可以参考本文档开启该功能。开启后会产生额外费用，详细收费标准请参见计费方式。如果您需要关闭该功能，请在DRDS控制台关闭。关闭后不再写入数据，可以查看历史数据，这部分历史数据会产生存储和索引费用。您可以在日志服务控制台删除Logstore以删除历史数据，删除后日志服务不再对该部分数据计费。

当前支持的区域：华东1、华东2、华北1、华北2、华南1。其它地区陆续开放中。

操作步骤

1. 登录*DRDS*控制台，并选择地域。
2. 在实例列表页面单击实例名称。
3. 在数据库列表页面单击数据库名称。
4. 在左侧导航栏中单击**SQL审计与分析**。

5. 请根据页面提示进行授权。授权后，DRDS有权限将日志数据写入日志服务。
6. 在**日志状态**一栏打开功能开关。



7. 确认是否导入历史数据。

   默认情况下，只对开启SQL日志审计功能之后产生的日志数据进行审计分析。

   如果您发现DRDS数据库的数据被修改，但是并未开启SQL审计功能，可以在开启功能时导入历史数据，将过去发生的日志加入审计分析范围，追溯数据篡改者。 DRDS 会根据您 DRDS 实例节点上的日志存储情况，动态检测支持导入的历史数据范围，目前最多支持导入七天内的数据。

   - 如果需要导入历史数据，请打开功能开关，并指定**回溯开始时间**和**回溯结束时间**，并单击**启用**。

     导入历史数据功能基于DRDS节点保存的日志时间动态检测，最大支持导入七天的历史数据。 导入过程涉及文件操作，需要一定的时间，请耐心等待。您可以在控制台右上角的任务列表中查看日志回溯的进度。
   - 如果不需要导入历史数据，请直接单击**启用**。



预期结果

SQL审计功能已成功开启，相同Region下的DRDS数据库的审计日志会写入同一个日志服务的Logstore中。同时，日志服务为您创建以下默认配置。

> **说明：**
> 请勿随意删除或修改日志服务为您默认创建的Project、Logstore、索引和仪表盘设置，日志服务会不定期更新与升级SQL日志审计功能，专属日志库的索引与默认报表也会自动更新。

表 5-12: 默认配置

默认配置项	配置内容
Project	默认为您创建Project`drds-audit-区域名-阿里云账户ID`，例如`drds-audit-cn-qingdao-12345678`。
Logstore	默认为您创建Logstore`drds-audit-log`。
地域	相同Region下的DRDS数据库的审计日志会写入同一个日志服务的Logstore中。 例如，华东一（杭州）地域中所有可用区的DRDS数据库审计日志都会写入日志服务的Logstore`drds-log-cn-hangzhou-阿里云账户ID`中。
Shard	默认为您创建5个Shard，并开启自动分裂*Shard* 功能。

默认配置项	配置内容
索引	日志服务自动为您创建索引。您的索引设置会随着功能升级自动更新，请勿手动修改或删除索引。
日志存储时间	默认保存30天。30天以上的日志自动删除。 若您需要更长的存储时间，可以自定义修改。详细步骤请参考高级管理。
仪表盘	默认创建三个仪表盘，分别为： • 运营中心 • 性能中心 • 安全中心 关于仪表盘的更多信息，请参考日志报表。 说明： 请勿修改默认的仪表盘配置，默认仪表盘会随着功能升级自动更新。您可以参考仪表盘另外创建新的仪表盘。

5.8.3 日志字段

DRDS SQL审计日志包含的日志字段下表所示。

字段名称	字段说明	支持版本
__topic__	日志主题，格式为`drds_audit_log_{instance_id}_{db_name}`，例如如`drds_audit_log_drdsxyzabcd_demo_drds_db`。	所有版本
instance_id	DRDS实例ID。	所有版本
db_name	DRDS 数据库名。	所有版本
user	执行SQL的用户名。	所有版本
client_ip	访问DRDS实例的客户端IP。	所有版本
client_port	访问DRDS实例的客户端端口。	所有版本
sql	执行的SQL语句。	所有版本
trace_id	SQL执行的 TRACE ID。如果是事务的话， 会以跟踪ID加`-`作为前缀+数字表示，例如`drdsabcdxyz-1`，`drdsabcdxyz-2`等。	所有版本

字段名称	字段说明	支持版本
sql_code	模板SQL 的 HASH 值。	所有版本
hint	SQL执行的 HINT。	所有版本
table_name	查询涉及的表名，多表之间以逗号分隔。	所有版本
sql_type	SQL类型。包括：Select、Insert、Update、Delete、Set、Alter、Create、Drop、Truncate、Replace和Other。	所有版本
sql_type_detail	SQL 解析器的名称。	所有版本
response_time	响应时间，单位为ms。	5.3.4-15378085及之后版本支持
affect_rows	SQL执行返回行数，增删改时表示影响的行数，查询语句表示返回的行数。	5.3.4-15378085及之后版本支持
fail	SQL 执行是否出错。 • 0：成功 • 1：失败	5.3.4-15378085及之后版本支持

5.8.4 日志分析

DRDS SQL审计与分析，依托日志服务产品，提供强大的日志分析能力。本文档提供常见场景的SQL日志分析语句及示例。

开启DRDS SQL审计与分析功能之后，您可以在当前页面通过日志服务的查询分析语法进行SQL审计与分析。结合日志服务的查询分析语法，在日志分析页面，您可以快速定位问题SQL，并针对DRDS 数据库的SQL执行状况、性能指标，安全问题进行分析。日志服务的查询分析语法请参考：

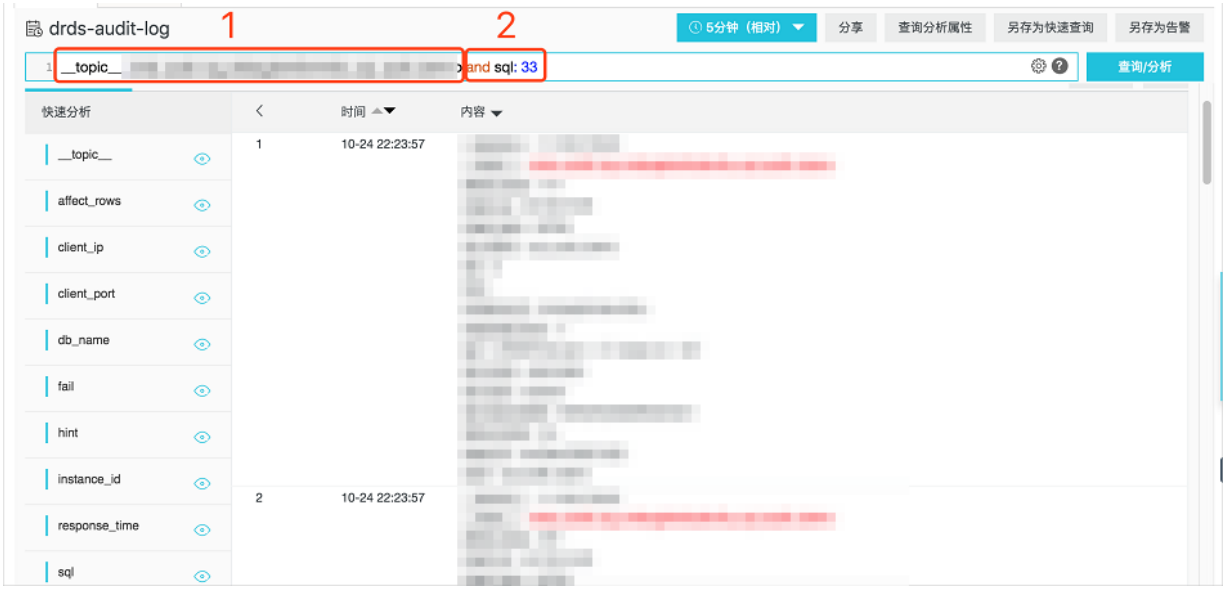
- 查询语法
- 实时分析简介

注意事项

相同Region下，DRDS数据库的审计日志都是写入同一个日志服务的Logstore里，所以DRDS SQL审计与分析的搜索页面会默认为您带上按照`__topic__`的过滤条件，保证您搜索到的SQL日志是当前DRDS数据库的。因此本文提供的所有的查询语句，都需要在已有的过滤条件后追加使用。

例如下图中：

- 序号1部分的语句为默认过滤条件。
- 序号2部分的语句为追加的过滤条件。



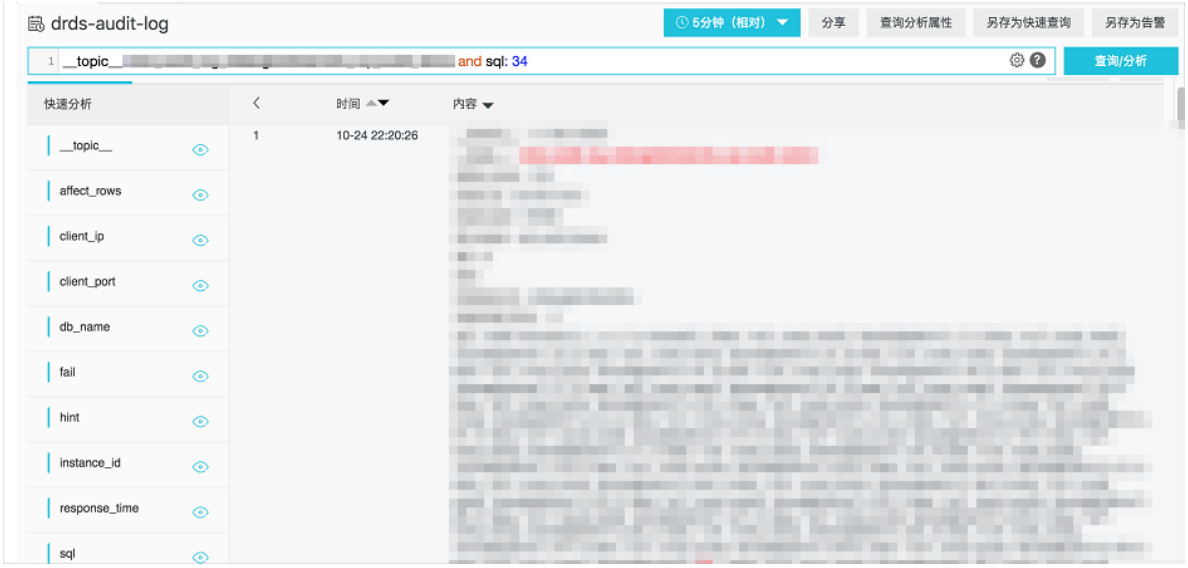
通过SQL快速定位问题

- 模糊搜索

  例如，查询包含“34”关键字的SQL语句，请在查询框中输入：

  ```
  and sql: 34
  ```

  查询结果如下图所示：



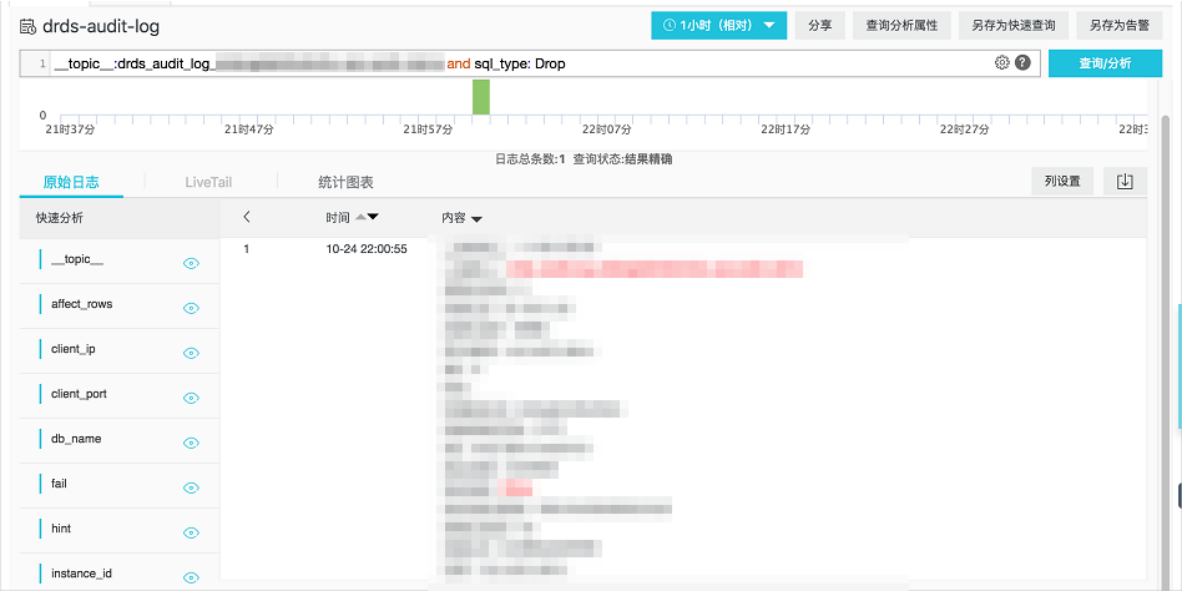
- 字段搜索

依赖预置的索引字段，DRDS SQL审计还支持根据字段搜索。

例如，查询Drop类型的SQL：

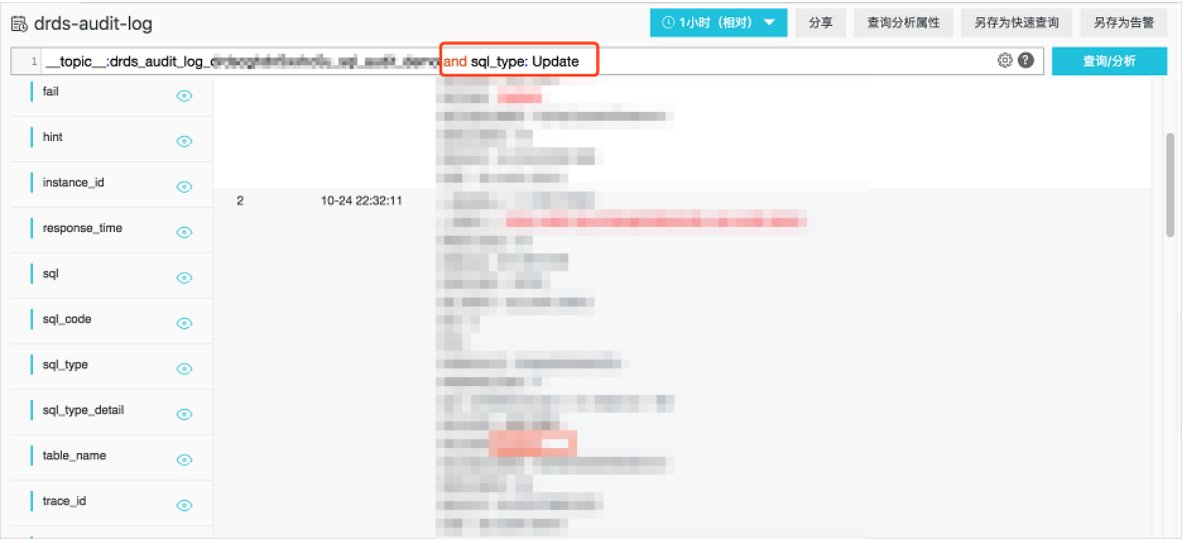
```
and sql_type:Drop
```

查询结果如下图所示：



说明：

日志服务支持鼠标点击自动生成查询语句，如下图所示。



- 多条件搜索

  通过“and”，“or”这类关键字，可以实现多条件的搜索。

例如，查询对id = 34 行的删除操作：

```
and sql:34 and sql_type: Delete
```

- 数值比较搜索

  索引字段中的"affect_rows", "response_time" 是数值类型，支持比较操作符。

  例如，查询response_time 大于1s 的 Insert SQL：

  ```
  and response_time > 1507 and sql_type: Insert
  ```

  例如，查询删除100行以上数据的SQL：

  ```
  and affect_rows > 100 and sql_type: Delete
  ```

SQL执行状况分析

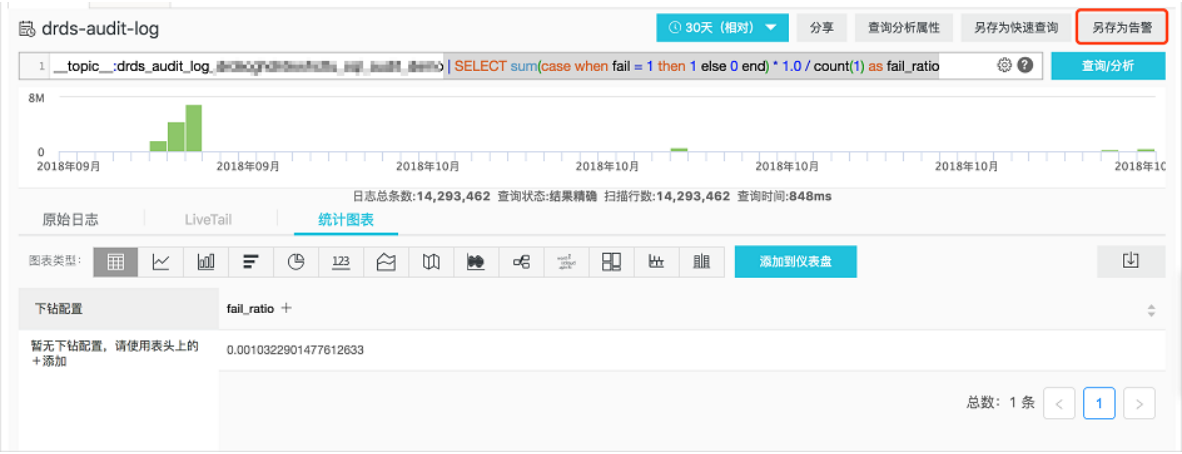
本节主要介绍DRDS数据执行状况相关的查询语句。

- **SQL执行失败率**

  通过以下语句查询SQL执行的失败率：

  ```
  | SELECT sum(case when fail = 1 then 1 else 0 end) * 1.0 / count(1)
  as fail_ratio
  ```

  查询结果如下图所示。



如果您的业务对SQL错误率敏感，可以在此查询结果的基础上，定制报警信息，单击下图中1所示的另存为告警。以下报警设置表示每隔15分钟，检查15分钟内SQL执行的错误率大于0.01的日志数量。您也可以根据业务需要定制告警。



- **SQL累计查询行数**

  通过以下语句查询Select 语句累计查询的行数：

  ```
  and sql_type: Select | SELECT sum(affect_rows)
  ```

- **SQL类型分布**

  通过以下语句查询SQL类型分布：

  ```
  | SELECT sql_type, count(sql) as times GROUP BY sql_type
  ```

- **SQL 独立用户IP分布**

  通过以下语句查询SQL 独立用户的IP地址分布：

  ```
  | SELECT user, client_ip, count(sql) as times GROUP BY user,
  client_ip
  ```

SQL性能分析

本节将给出典型的SQL性能分析的查询语句。

- **SELECT平均耗时**

通过以下语句查询SELECT语句的平均耗时：

```
and sql_type: Select | SELECT avg(response_time)
```

- **SQL执行耗时分布**

  通过以下语句查询SQL执行耗时分布情况：

```
and response_time > 0 | select case when response_time <= 10 then
 '<=10毫秒' when response_time > 10 and response_time <= 100 then '
10~100毫秒' when response_time > 100 and response_time <= 1000 then
 '100毫秒~1秒' when response_time > 1000 and response_time <= 10000
then '1秒~10秒' when response_time > 10000 and response_time <= 60000
 then '10秒~1分钟' else '>1分钟' end as latency_type, count(1) as cnt
group by latency_type order by latency_type DESC
```

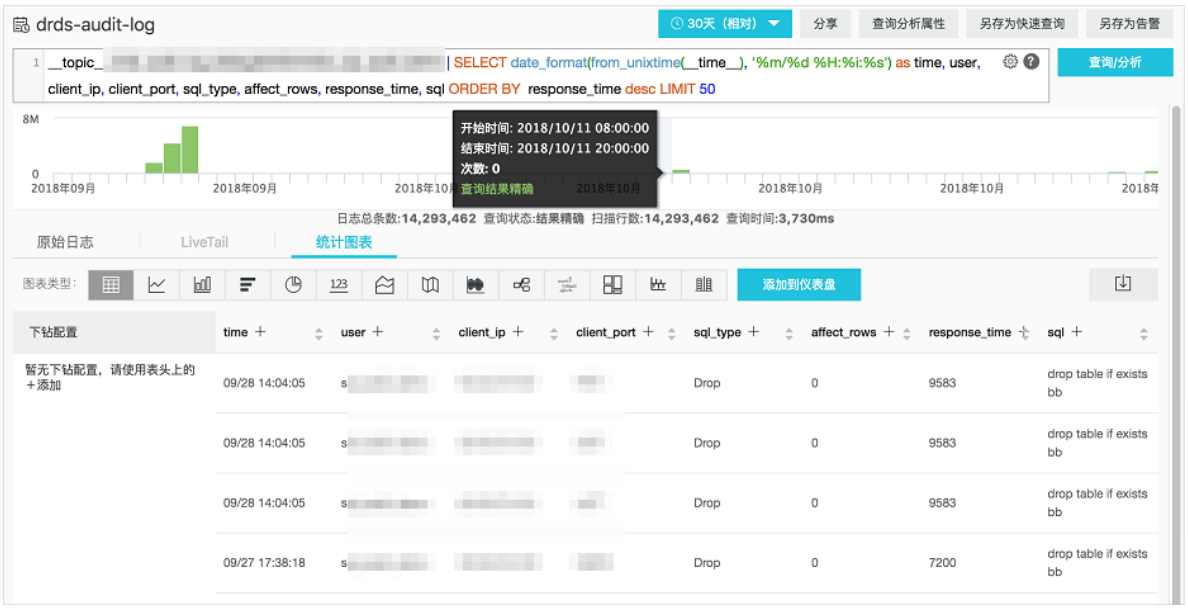
  以上语句分析指定时间段的SQL执行时间分布，您也可以调整时间段的范围，以获取更加精细的结果。

- **慢SQL Top 50**

  通过以下语句查询系统慢SQL的列表：

```
| SELECT date_format(from_unixtime(__time__), '%m/%d %H:%i:%s')
as time, user, client_ip, client_port, sql_type, affect_rows,
response_time, sql ORDER BY response_time desc LIMIT 50
```

  查询结果如下图所示，结果中包含SQL执行时间、执行的用户名、IP地址、端口号、SQL类型、影响行数、执行时间以及SQL的文本。



- **高代价SQL模板 Top 10**

大多数应用中，SQL通常基于若干模板动态生成的，只是参数不同。可以根据模板ID，找到应用中高代价的SQL模板，进行分析优化。

输入如下的查询语句：

```
| SELECT sql_code as "SQL模板ID", round(total_time * 1.0 /sum(
total_time) over() * 100, 2) as "总体耗时比例(%)" ,execute_times as
 "执行次数", round(avg_time) as "平均执行时间",round(avg_rows) as "平
均影响行数", CASE WHEN length(sql) > 200 THEN concat(substr(sql, 1,
200), '......') ELSE trim(lpad(sql, 200, ' ')) end as "样例SQL" FROM
 (SELECT sql_code, count(1) as execute_times, sum(response_time)
as total_time, avg(response_time) as avg_time, avg(affect_rows) as
avg_rows, arbitrary(sql) as sql FROM log GROUP BY sql_code) ORDER BY
 "总体耗时比例(%)" desc limit 10
```

SQL模板ID	总体耗时比例(%)	执行次数	平均执行时间	平均影响行数	样例SQL
ac0c0ca2	34.97	38116	10343.0	1.0	
4f3b8c1d	27.22	30212	10157.0	8.0	

统计结果包括SQL模板ID，该模板SQL占总体SQL的耗时比例，执行次数，平均执行时间，平均影响行数，以及样例SQL。为了显示效果，该列按照200的长度截断。上述查询是按照总体耗时比例排序，当然您也可以根据平均执行时间，执行次数进行排序，排查问题。

- **事务平均执行时长**

  对于相同事务内的SQL，预置的trace_id字段前缀相同，后缀为'-' + 序号；非事务的SQL的trace_id中则不包含'-'。基于此，我们可以对事务的性能进行相关分析。

  > **说明：**
  > 事务分析由于涉及前缀匹配操作，查询效率会低于其它类型的查询操作。

  例如，通过以下语句查询事务的平均执行耗时：

  ```
  | SELECT sum(response_time) / COUNT(DISTINCT substr(trace_id, 1,
  strpos(trace_id, '-') - 1)) where strpos(trace_id, '-') > 0
  ```

- **慢事务 Top 10**

按照事务的执行时间排序，可以查询慢事务的列表，查询语句如下：

```
| SELECT substr(trace_id, 1, strpos(trace_id, '-') - 1) as "事务ID
" , sum(response_time) as "事务耗时" where strpos(trace_id, '-') > 0
GROUP BY substr(trace_id, 1, strpos(trace_id, '-') - 1) ORDER BY "事
务耗时" DESC LIMIT 10
```

事务ID	事务耗时
db3226a20402000	827632
db323335f000000	415490
db322833c801000	388104
db322893c401000	381963
db3228bf8c01000	379173
db322928e001000	372432
db3235851402000	305926

在此基础上，可以根据查到的慢事务ID，搜索该事务下的所有SQL，分析执行慢的具体原因，查询语句如下：

```
and trace_id: db3226a20402000*
```

- 大批量操作事务 **Top10**

  按照事务内SQL影响的行数，可以获取大批量操作的事务列表，查询语句如下：

  ```
  | SELECT substr(trace_id, 1, strpos(trace_id, '-') - 1) as "事务ID
  " , sum(affect_rows) as "影响行数" where strpos(trace_id, '-') > 0
  GROUP BY substr(trace_id, 1, strpos(trace_id, '-') - 1) ORDER BY "影
  响行数" DESC LIMIT 10
  ```

SQL安全性分析

以下为典型SQL安全性分析的查询语句。

- 错误**SQL**类型分布

  ```
  and fail > 0 | select sql_type, count(1) as "错误次数" group by
  sql_type
  ```

- 高危**SQL**列表

高危SQL是Drop 或者 Truncate类型的SQL，您也可以根据需求增加更多条件。

```
and sql_type: Drop OR sql_type: Truncate
```

- 大批量删除SQL列表

```
and affect_rows > 100 and sql_type: Delete | SELECT date_format(
from_unixtime(__time__), '%m/%d %H:%i:%s') as time, user, client_ip
, client_port, affect_rows, sql ORDER BY affect_rows desc LIMIT 50
```

5.8.5 日志报表

DRDS SQL审计与分析，依托日志服务，为您提供开箱即用的报表，包括运营中心、性能中心、安全中心，让您对DRDS数据库的执行状况、性能指标、潜在安全问题了如指掌。

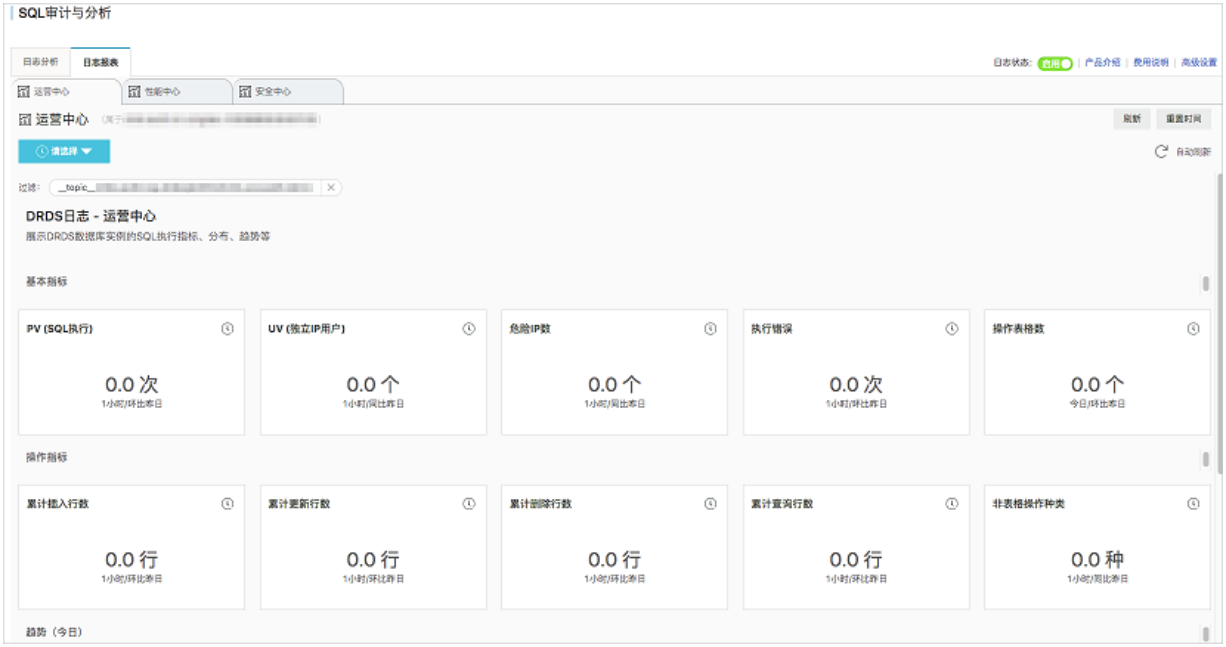
开启SQL日志审计之后，在当前页面单击进入日志报表页签，即可查看日志服务提供的报表页面，包括运营中心、性能中心和安全中心。



说明：

- 由于相同地区的DRDS数据库的审计日志均写入日志服务同一个Logstore中，查看当前DRDS数据库的报表数据时，默认为您添加基于`__topic__:drds_audit_log_实例ID_数据库名`的过滤条件，表示查看当前数据库的数据。例如，`drds_audit_log_drdsxyzabcd_demo_drds_db`。
- DRDS实例版本如果早于版本5.3.4-15378085，SQL日志中会缺少相关字段（日志字段说明请参考日志字段），日志报表页面只提供简化版的运营中心。如需使用完整版的报表，请升级到最新版本。

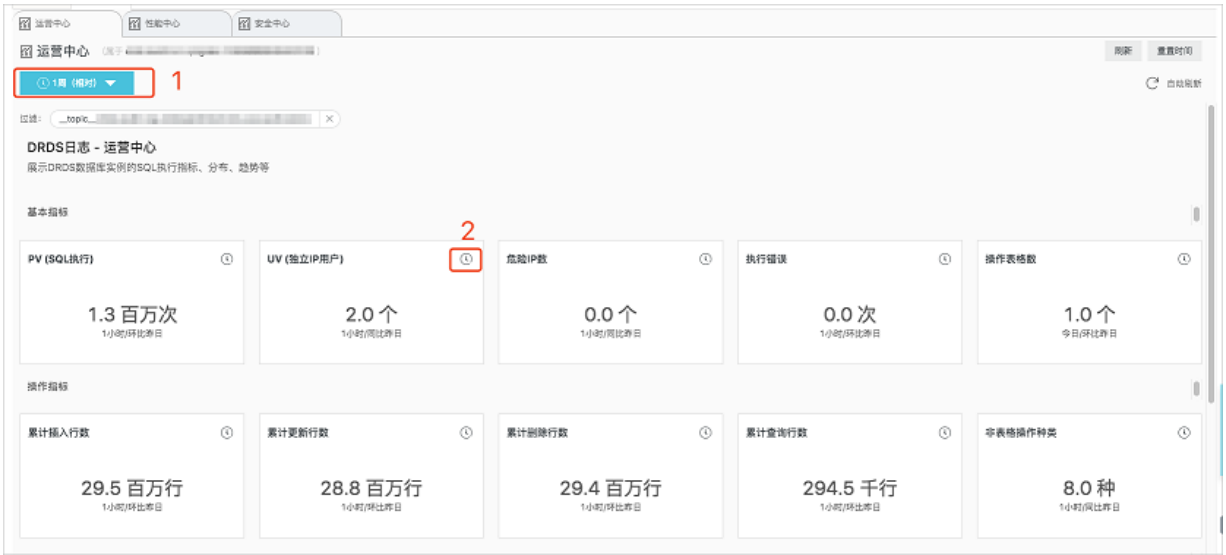
图 **5-51:** 查看报表



日志报表页面的所有图表都是基于不同时间段的数据统计结果，您可以根据需求修改时间范围页。时间范围的修改既可面向所有图表，也可以针对单一图表。

- 单击时间选择器（图中1位置），可以在弹出的时间选择控件中修改当前页面所有图表的时间范围。
- 单击图表的时间选择器（图中2位置），可以修改当前图表的时间范围。

图 **5-52:** 时间选择器



运营中心

运营中心展示DRDS数据库的SQL执行指标、分布、趋势等。

图表	类型	默认时间范围	说明
PV（SQL 执行）	单值	1小时（相对）	SQL执行的次数
UV（独立IP用户）	单值	1小时（相对）	独立的用户及IP数目
危险IP数	单值	1小时（相对）	危险IP的数目，危险IP的定义详见：安全检测函数
执行错误	单值	1小时（相对）	执行错误的SQL数目
操作表格数	单值	1小时（相对）	SQL操作的表格总数
累计插入行数	单值	1小时（相对）	插入操作累计插入的数据行数
累计更新行数	单值	1小时（相对）	更新操作累计更新的数据行数
累计删除行数	单值	1小时（相对）	删除操作累计删除的数据行数
累计查询行数	单值	1小时（相对）	查询操作累计返回的数据行数
非表格操作种类	单值	1小时（相对）	非表格操作的SQL种类，例如show variables like
SQL执行趋势	柱状图	1小时（相对）	SQL 执行的趋势分布以及对应的错误SQL的分布趋势
操作表格	流图	1小时（相对）	SQL 操作表格的分布情况
SQL类型	流图	1小时（相对）	SQL类型的按照时间的分布情况
操作用户分布	饼图	1小时（相对）	执行SQL用户的分布情况
SQL执行类型分布	面积图	1小时（相对）	当前时间范围内SQL类型的比例
操作最多的表格Top 50	表格	1小时（相对）	操作最多的表格列表，包括表格的名称以及对应的读、删、改、插的次数
执行分布（世界）	地图	1小时（相对）	执行SQL的client ip在世界地图上的分布情况
执行分布（中国）	地图	1小时（相对）	执行SQL的client ip在中国地图上的分布情况

性能中心

性能中心展示DRDS数据库的性能指标、快慢分布、慢SQL、高代价SQL的具体分布与来源等。

图表	类型	默认时间范围	说明
SQL执行峰值	单值	1小时（相对）	每秒SQL执行条数的峰值
查询带宽峰值	单值	1小时（相对）	每秒查询SQL返回行数的峰值
插入带宽峰值	单值	1小时（相对）	每秒插入SQL插入的行数峰值
更新带宽峰值	单值	1小时（相对）	每秒更新SQL更新的行数峰值
删除带宽峰值	单值	1小时（相对）	每秒删除SQL删除的行数峰值
平均执行时间	单值	1小时（相对）	SQL平均的执行时间
查询SQL	单值	1小时（相对）	平均每秒查询SQL执行的条数
插入SQL	单值	1小时（相对）	平均每秒插入SQL执行的条数
更新SQL	单值	1小时（相对）	平均每秒更新SQL执行的条数
删除SQL	单值	1小时（相对）	平均每秒删除SQL执行的条数
查询更新带宽趋势	折线图	1小时（相对）	查询SQL、更新SQL操作行数随时间的分布情况
SQL执行时间分布	饼图	1小时（相对）	SQL执行时间的分布情况
慢SQL表格分布	饼图	1小时（相对）	慢SQL（执行时间超过1s）的表格分布情况
慢SQL用户分布	饼图	1小时（相对）	慢SQL（执行时间超过1s）的用户分布情况
慢SQL类型分布	饼图	1小时（相对）	慢SQL（执行时间超过1s）的类型分布情况
慢SQL列表 Top 50	表格	1小时（相对）	慢SQL（执行时间超过1s）的列表，包括：时间、客户端、时间、DRDS实例、数据库、表格、用户、影响行数、SQL类型、SQL文本
SQL模板执行时间Top20	表格	1小时（相对）	按照SQL模板统计该模板SQL的执行情况，包括：SQL模板ID、总体耗时比例、执行次数、平均执行时间、平均影响行数、样例SQL

图表	类型	默认时间范围	说明
事务执行影响行数Top20	表格	1小时（相对）	事务影响行数的Top20列表，包括：事务ID、影响行数
事务执行时间Top20	表格	1小时（相对）	事务执行时间的Top20列表，包括：事务ID、影响行数

安全中心

安全中心展示DRDS数据库的失败SQL、危险SQL和大批量删除、修改事件的详情、分布和趋势等。

图表	类型	默认时间范围	说明
错误数	单值	1小时（相对）	失败SQL的执行次数
大批量删除事件	单值	1小时（相对）	大批量删除事件（超过100行）的SQL数目
大批量修改事件	单值	1小时（相对）	大批量修改事件（超过100行）的SQL数
危险SQL执行	单值	1小时（相对）	危险SQL（Drop、Truncate操作）的数目
危险IP数	单值	1小时（相对）	危险IP的数目，危险IP的定义详见：安全检测函数
错误操作类型分布	面积图	1小时（相对）	失败SQL的类型分布
出错客户端外网分布	地图	1小时（相对）	失败SQL的客户端在中国地图的分布
错误最多的客户端	表格	1小时（相对）	失败SQL的客户端列表，包括IP，错误数目、错误SQL类型、出错SQL样例
危险SQL执行列表	表格	1小时（相对）	危险SQL的列表，包括时间、IP、SQL、DRDS 实例ID，数据库、表格、用户
大批量删除事件Top 50	表格	1小时（相对）	大批量删除SQL的列表，包括最早执行时间、最近执行时间、DRDS实例ID、数据库、表格、执行次数、平均删除行数、平均时长、样例SQL

图表	类型	默认时间范围	说明
大批量修改事件 Top 50	表格	1小时（相对）	大批量修改SQL的列表，包括最早执行时间、最近执行时间、DRDS实例ID、数据库、表格、执行次数、平均修改行数、平均时长、样例SQL

5.8.6 高级管理

DRDS SQL审计与分析支持高级管理，您可以通过高级管理跳转到日志服务控制台，修改SQL日志的存储时间、对SQL日志进实时订阅与消费、数据投递和对接其他可视化等高级操作。

开启SQL日志审计后，在当前页面右上角单击高级管理可以跳转到日志服务控制台，修改日志存储时间、配置日志消费等高级操作。

> **说明：**
> 由于相同地区DRDS数据库的SQL审计日志存储在同一Logstore中，如果修改了Logstore的审计日志存储时间，该修改会对该地域下所有DRDS数据库生效。即仅支持修改同一地域所有数据库的日志保存时间，不支持修改某一数据库的日志保存时间。

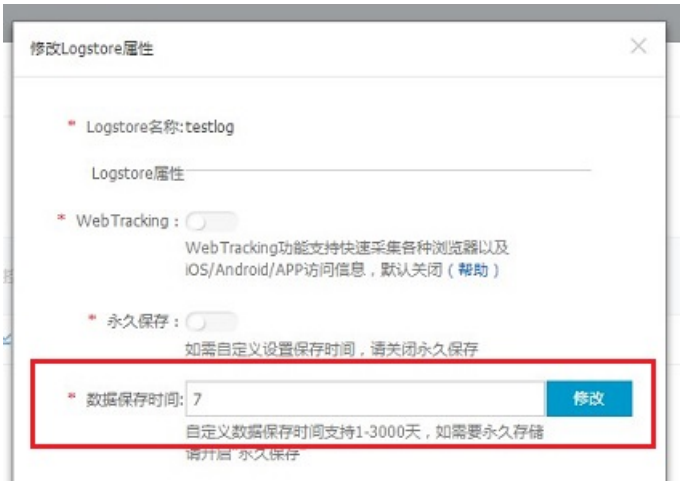


修改日志保存时间

1. 开启SQL日志审计后，在当前页面右上角单击高级管理跳转到日志服务控制台。
2. 单击操作列的修改。

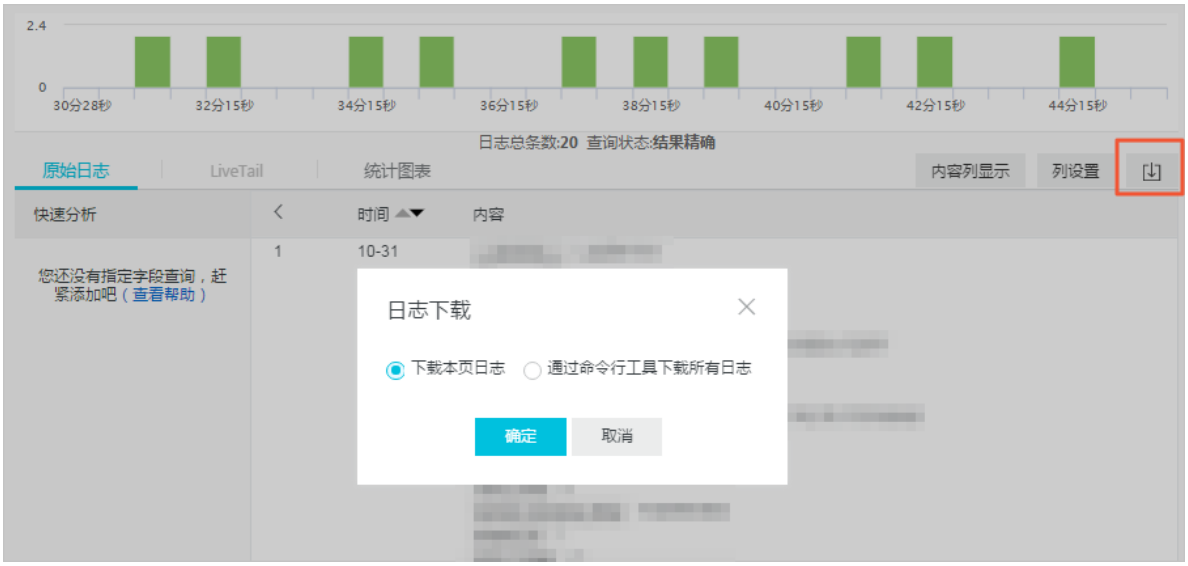
3. 在弹出页面中修改数据保存时间，并单击修改。

图 5-53: 修改日志保存时间



导出日志数据

1. 开启SQL日志审计后，单击原始日志页签右侧的日志下载按钮打开日志下载对话框。



2. 在日志下载对话框中单击下载本页日志以CSV格式将本页面的日志保存到本地。
3. 您也可以单击通过命令行工具下载所有日志下载所有日志。



a. 单击下载日志对话框中的命令行工具*CLI*用户手册。

b. 安装命令行工具。

c. 单击安全信息管理页面链接查看并复制当前用户的秘钥ID和KEY。

d. 单击复制命令行并用当前用户的秘钥ID和KEY替换该命令行中`【步骤2中的秘钥ID】`和`【步骤2中的秘钥Key】`。

e. 在CLI命令行工具中执行该命令。

命令执行后，DRDS SQL审计日志将将自动下载并保存到运行命令的当前目录下的**download_data.txt**文件中。

为子账号授权

请参考子账号授权为子账号赋予开启或操作SQL审计分析功能的权限。

其他高级操作

- 告警与通知
- 实时订阅与消费
- 数据投递
- 对接其他可视化工具

5.8.7 为子账号授予DRDS SQL审计权限

子账号开通或使用DRDS SQL审计功能之前，需要由主账号为其授权。

背景信息

使用日志服务实时日志分析，需要如下权限 ：

操作类型	支持的操作账号类型
开通日志服务（全局一次性操作）	主账号
授权DRDS写入数据到您的日志服务 （全局一次性操作）	• 主账号 • 具备AliyunLogFullAccess权限的子账号 • 具备指定权限的子账号
使用日志分析功能	• 主账号 • 具备AliyunLogFullAccess权限的子账号 • 具备指定权限的子账号

您也可以根据需求为子账号授权。

- 为子账号授予日志服务的全部操作权限：授予全部管理权限AliyunLogFullAccess。详细步骤请参考授权RAM 用户。
- 主账号开启DRDS SQL审计分析后，为子账号授予日志查看的权限：授予只读权限AliyunLogReadOnlyAccess。详细步骤请参考授权RAM 用户。
- 仅为子账号授予开通及及使用DRDS SQL审计与分析的权限，不授予其他日志服务管理权限：创建自定义授权策略，并为子账号授予该自定义授权策略。详细步骤请参考本文档。

操作步骤

1. 登录 *RAM* 控制台。
2. 在策略管理中打开自定义授权策略页签。
3. 在页面右上角单击新建授权策略。
4. 单击空白模板，并输入策略名称和策略内容。

   请替换参数后，输入以下策略内容。您也可以添加其他自定义授权内容。

   说明：

   请将${Project}与${Logstore}替换为您的日志服务Project和Logstore名称。

```
{
  "Version": "1",
  "Statement": [
   {
      "Action": "log:GetProject",
      "Resource": "acs:log:*:*:project/${Project}",
      "Effect": "Allow"
    },
    {
      "Action": "log:CreateProject",
      "Resource": "acs:log:*:*:project/*",
      "Effect": "Allow"
    },
 {
      "Action": "log:ListLogStores",
      "Resource": "acs:log:*:*:project/${Project}/logstore/*",
      "Effect": "Allow"
    },
    {
      "Action": "log:CreateLogStore",
      "Resource": "acs:log:*:*:project/${Project}/logstore/*",
      "Effect": "Allow"
    },
 {
      "Action": "log:GetIndex",
      "Resource": "acs:log:*:*:project/${Project}/logstore/${
Logstore}",
      "Effect": "Allow"
    },
    {
      "Action": "log:CreateIndex",
      "Resource": "acs:log:*:*:project/${Project}/logstore/${
Logstore}",
      "Effect": "Allow"
    },
    {
      "Action": "log:UpdateIndex",
      "Resource": "acs:log:*:*:project/${Project}/logstore/${
Logstore}",
      "Effect": "Allow"
    },
    {
      "Action": "log:CreateDashboard",
      "Resource": "acs:log:*:*:project/${Project}/dashboard/*",
```

```
      "Effect": "Allow"
    },
 {
      "Action": "log:UpdateDashboard",
      "Resource": "acs:log:*:*:project/${Project}/dashboard/*",
      "Effect": "Allow"
    },
 {
      "Action": "log:CreateSavedSearch",
      "Resource": "acs:log:*:*:project/${Project}/savedsearch/*",
      "Effect": "Allow"
    },
 {
      "Action": "log:UpdateSavedSearch",
      "Resource": "acs:log:*:*:project/${Project}/savedsearch/*",
      "Effect": "Allow"
    }
  ]
}
```

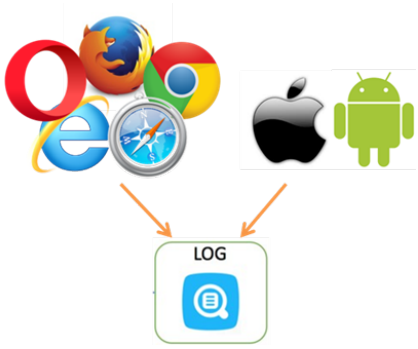


5. 单击新建授权策略。
6. 在左侧导航栏中单击用户管理。
7. 找到子账号并单击对应的授权。
8. 添加上文中创建的自定义授权策略，并单击确定。

6 其他采集方式

6.1 Web Tracking

日志服务支持通过Web Tracking功能进行HTML、H5、iOS和 Android平台日志数据的采集，支持自定义维度和指标。



如上图所示，使用Web Tracking功能可以采集各种浏览器以及iOS、Android APP的用户信息（除*iOS/Android SDK*外），例如：

- 用户使用的浏览器、操作系统、分辨率等。
- 用户浏览行为记录，比如用户网站上的点击行为、购买行为等。
- 用户在APP中停留时间、是否活跃等。

注意事项

- 使用Web Tracking意味着该Logstore打开互联网匿名写入的权限，没有经过有效鉴权，可能会产生脏数据。
- 仅支持Get请求，不支持POST请求；且不支持上传16KB以上的body。

配置步骤

步骤1 开通Web Tracking

您可以通过控制台或SDK方式开通Web Tracking。

- **通过控制台开通Web Tracking**

  1. 在Logstore列表页面，选中需要开通Web Tracking功能的Logstore，单击右侧的修改。
  2. 打开 Web Tracking 开关。



- 通过 **Java SDK** 开通**Web Tracking**

  使用 *JAVA SDK*：

```
import com.aliyun.openservices.log.Client;
import com.aliyun.openservices.log.common.LogStore;
import com.aliyun.openservices.log.exception.LogException;
public class WebTracking {
  static private String accessId = "your accesskey id";
  static private String accessKey = "your accesskey";
  static private String project = "your project";
  static private String host = "log service data address";
  static private String logStore = "your logstore";
  static private Client client = new Client(host, accessId,
accessKey);
  public static void main(String[] args) {
      try {
          //在已经创建的Logstore 上开通Web Tracking功能。
          LogStore logSt = client.GetLogStore(project, logStore).
GetLogStore();
          client.UpdateLogStore(project, new LogStore(logStore,
logSt.GetTtl(), logSt.GetShardCount(), true));
          //关闭Web Tracking功能。
          //client.UpdateLogStore(project, new LogStore(logStore,
logSt.GetTtl(), logSt.GetShardCount(), false));
          //新建支持Web Tracking功能的Logstore。
          //client.UpdateLogStore(project, new LogStore(logStore, 1
, 1, true));
      }
      catch (LogException e){
          e.printStackTrace();
      }
  }
}
```

步骤2 收集日志数据

Logstore开通Web Tracking功能后，可以使用以下三种方法上传数据到Logstore中。

 说明：

建议您使用SDK方式上传日志数据。

- 使用 **js SDK**

1. 将 *loghub-tracking.js* 复制到 *web* 目录，并在页面中引入如下脚本：

   单击下载

   ```
   <script type="text/javascript" src="loghub-tracking.js" async></
   script>
   ```

   说明：

   为了不阻塞页面加载，脚本会异步发送 HTTP 请求，如果页面加载过程中需要多次发送数据，后面的请求会覆盖前面的 HTTP 请求，看到的现象是浏览器中会显示 Web Tracking 请求退出。使用同步发送可以避免该问题，同步发送请在脚本中执行如下语句替换：

   原始语句：

   ```
   this.httpRequest_.open("GET", url, true)
   ```

   替换最后一个参数变成同步发送：

   ```
   this.httpRequest_.open("GET", url, false)
   ```

2. 创建 Tracker 对象。

   ```
   var logger = new window.Tracker('${host}','${project}','${logstore
   }');
   logger.push('customer', 'zhangsan');
   logger.push('product', 'iphone 6s');
   logger.push('price', 5500);
   logger.logger();
   logger.push('customer', 'lisi');
   logger.push('product', 'ipod');
   logger.push('price', 3000);
   logger.logger();
   ```

   其中各个参数的含义如下：

字段	含义
`${host}`	您日志服务所在Region的endpoint。
`${project}`	您在日志服务中开通的Project名称。
`${logstore}`	`${project}`中的Logstore的名称。

执行以上命令后，可以在日志服务看到如下两条日志：

```
customer:zhangsan
product:iphone 6s
price:5500
```

```
customer:lisi
product:ipod
price:3000
```

- 使用**HTTP GET**请求

```
curl --request GET 'http://${project}.${host}/logstores/${logstore}/
track?APIVersion=0.6.0&key1=val1&key2=val2'
```

其中各个参数的含义如下：

字段	含义
${project}	您在日志服务中开通的Project名称。
${host}	您日志服务所在地区的域名。
${logstore}	`${project}` 下面开通Web Tracking功能的某一个Logstore的名称。
APIVersion=0.6.0	保留字段，必选。
__topic__=yourtopic	指定日志的topic，保留字段，可选
key1=val1、*key2=val2*	您要上传到日志服务的Key-Value对，可以有多个，但是要保证URL的长度小于16KB。

- 使用 **HTML img** 标签

```
<img src='http://${project}.${host}/logstores/${logstore}/track.gif?
APIVersion=0.6.0&key1=val1&key2=val2'/>
<img src='http://${project}.${host}/logstores/${logstore}/track_ua.
gif?APIVersion=0.6.0&key1=val1&key2=val2'/>
```

各个参数的含义同上，track_ua.gif除了将自定义的参数上传外，在服务端还会将http头中的UserAgent、referer也作为日志中的字段。



说明：

若您需要采集HTTPS页面的referer， 那么上述Web Tracking的链接也必须为HTTPS。

数据上传到日志服务之后，可以使用日志服务查询分析功能实时检索、分析日志数据，并通过多样的可视化方案展示实时分析结果。也可以使用日志服务提供的 *Consumer Library* 消费数据。

6.2 Logstash

6.2.1 安装

日志服务提供Logstash插件，支持通过Logstash上传日志数据。

背景信息

Logstash是一款流行的开源采集软件，您可以通过安装logstash-output-logservice插件上传数据到报日志服务。插件的GitHub项目地址：*Logstash插件*。

操作步骤

1. 安装Java。
   1. 下载安装包。

      请进入*Java 官网* 下载JDK并双击进行安装。
   2. 设置环境变量。

      打开高级系统设置，新增或修改环境变量。

      - **PATH:** *C:\Program Files\Java\jdk1.8.0_73\bin*
      - **CLASSPATH:** *C:\Program Files\Java\jdk1.8.0_73\lib;C:\Program Files\Java\jdk1.8.0_73\lib\tools.jar*
      - **JAVA_HOME:** *C:\Program Files\Java\jdk1.8.0_73*
   3. 验证。

      执行 `PowerShell` 或 `cmd.exe` 进行验证：

      ```
      PS C:\Users\Administrator> java -version
      java version "1.8.0_73"
      Java(TM) SE Runtime Environment (build 1.8.0_73-b02)
      Java HotSpot(TM) 64-Bit Server VM (build 25.73-b02, mixed mode)
      PS C:\Users\Administrator> javac -version
      javac 1.8.0_73
      ```
2. 安装Logstash。
   1. 下载安装包。

      官网下载：*Logstash主页* 。

      说明：

      - 建议下载Logstash 5.0及以上版本以运行插件。

- 经测试，在Mac OS 10.14.1、Windows 7、Linux（CentOS 7）三种系统环境下，Logstash 6.4.3版本安装插件正常、工作正常。

2. 安装。

   解压安装包到安装目录。

3. 安装Logstash写日志服务插件。

   请根据机器所处网络环境决定在线或离线安装模式：

   - 在线安装。

     该插件托管于 RubyGems，更多信息请单击查看。

     执行 `PowerShell` 或 `cmd.exe`，进入Logstash安装目录。执行以下命令安装Logstash：

     ```
     PS C:\logstash-6.4.3> .\bin\plugin install logstash-output-
     logservice
     ```

   - 离线安装。

     官网下载：进入 *logstash-output-logservice* 页面，单击右下角 下载 按钮。

     如采集日志机器无法访问公网，请拷贝下载的 gem 包到本地目录。执行 *`PowerShell`* 或 *`cmd.exe`*，进入 Logstash 安装目录。执行以下命令安装Logstash：

     ```
     PS C:\logstash-6.4.3> .\bin\plugin install C:\logstash-6.4.3\
     logstash-output-logservice-0.4.0.gem
     ```

   - 验证。

     ```
     PS C:\logstash-6.4.3> .\bin\plugin list
     ```

     在本机已安装的插件列表中可以找到 logstash-output-logservice。

6.2.2 配置 Logstash 为 Windows Service

在 PowerShell 下启动 *`logstash.bat`*，Logstash进程会在前台工作，一般用于配置测试和采集调试。建议调试通过后把Logstash设置为 Windows Service，可以保持后台运行以及开机自启动。

除了将Logstash设置为Windows Service之外，您还可以通过命令行启动、停止、修改和删除服务。更多 NSSM 使用方法请参考 *NSSM*官方文档。

添加服务

一般用于首次部署时执行，如已添加过服务，请跳过该步骤。

您可以执行以下命令添加服务。

- 32 位系统

```
C:\logstash-2.2.2-win\nssm-2.24\win32\nssm.exe install logstash "C:\
logstash-2.2.2-win\bin\logstash.bat" "agent -f C:\logstash-2.2.2-win
\conf"
```

- 64 位系统

```
C:\logstash-2.2.2-win\nssm-2.24\win64\nssm.exe install logstash "C:\
logstash-2.2.2-win\bin\logstash.bat" "agent -f C:\logstash-2.2.2-win
\conf"
```

启动服务

如Logstash *conf* 目录后有配置文件更新，请先停止服务，再启动服务。

您可以执行以下命令启动服务。

- 32 位系统

```
C:\logstash-2.2.2-win\nssm-2.24\win32\nssm.exe start logstash
```

- 64 位系统

```
C:\logstash-2.2.2-win\nssm-2.24\win64\nssm.exe start logstash
```

停止服务

您可以执行以下命令停止服务。

- 32 位系统

```
C:\logstash-2.2.2-win\nssm-2.24\win32\nssm.exe stop logstash
```

- 64 位系统

```
C:\logstash-2.2.2-win\nssm-2.24\win64\nssm.exe stop logstash
```

修改服务

您可以执行以下命令修改服务。

- 32 位系统

```
C:\logstash-2.2.2-win\nssm-2.24\win32\nssm.exe edit logstash
```

- 64 位系统

```
C:\logstash-2.2.2-win\nssm-2.24\win64\nssm.exe edit logstash
```

删除服务

您可以执行以下命令删除服务。

- 32 位系统

```
C:\logstash-2.2.2-win\nssm-2.24\win32\nssm.exe remove logstash
```

- 64 位系统

```
C:\logstash-2.2.2-win\nssm-2.24\win64\nssm.exe remove logstash
```

6.2.3 创建 Logstash 采集配置

背景信息

相关插件

- **logstash-input-file**

  通过该插件tail方式收集日志文件，详细信息请参考 *logstash-input-file*。

  说明：

  path 填写文件路径时请使用UNIX 模式的分隔符，如：*C:/test/multiline/*.log*，否则无法支持模糊匹配。

- **logstash-output-logservice**

  通过该插件可以 input 插件采集的日志写出到日志服务。

参数	说明
endpoint	日志服务入口。如 *http://regionid.example.com*。
project	日志服务项目名称。
logstore	日志库名称。
topic	日志主题名称，默认设置空即可。
source	日志来源，如为空则自动取本机 IP，否则以设置值为准。

参数	说明
access_key_id	阿里云云账号秘钥 ID。
access_key_secret	阿里云云账号秘钥 secret。
max_send_retry	数据包发送日志服务发生异常时最大重试次数，重试不成功的数据包丢弃，重试间隔为 200 毫秒。

操作步骤

1. 创建采集配置。

   新增配置文件到 *C:\logstash-2.2.2-win\conf* 目录后，重启Logstash生效。

   可以为每种日志新建一个配置文件，格式为 **.conf*。建议统一保存于 *C:\logstash-2.2.2-win\conf* 目录下，以方便管理。

   说明：

   配置文件格式必须以 UTF-8 无 BOM 格式编码，可以通过 notepad++ 修改文件编码格式。

   - IIS 日志

     请参考 *Logstash 收集 IIS 日志*。

   - CSV 日志

     使用采集日志的系统时间作为上传日志时间，请参考*Logstash 收集 CSV 日志*。

   - 自带时间日志

     以CSV日志格式为例，以日志内容中的时间作为上传日志时间，请参考*Logstash 收集 CSV 日志*。

   - 通用日志

     默认使用采集日志的系统时间作为上传的日志时间，日志不解析字段，支持单行、多行日志格式。请参考 *Logstash 收集其它日志*。

2. 验证配置语法。

1. 执行`PowerShell`或`cmd.exe`，进入Logstash安装目录。执行以下命令验证配置。

```
PS C:\logstash-2.2.2-win\bin> .\logstash.bat agent --configtest --
config C:\logstash-2.2.2-win\conf\iis_log.conf
```

2. 修改收集配置文件，在 output 阶段临时添加一行 rubydebug 配置以输出采集结果到控制台。

   配置中 type 字段请自行设置。

```
output {
if [type] == "***" {
  stdout { codec => rubydebug }
  logservice {
  ...
  }
 }
}
```

3. 执行`PowerShell`或`cmd.exe`，进入Logstash安装目录启动进程。执行以下命令：

```
PS C:\logstash-2.2.2-win\bin> .\logstash.bat agent -f C:\logstash-
2.2.2-win\conf
```

验证完成后，请结束 *`logstash.bat`* 进程并删除 rubydebug 临时配置项。

后续操作

在 PowerShell 下启动 *`logstash.bat`*，logstash 进程会在前台工作，一般用于配置测试和采集调试。建议调试通过后把Logstash设置为 Windows Service，可以保持后台运行以及开机自启动。有关如何将Logstash设置为 Windows Service，参见 配置 *Logstash* 为 *Windows Service*。

6.2.4 进阶功能

Logstash 提供了 大量插件，可以满足个性化需求，例如：

- *grok*：通过正则表达式结构解析日志内容成多个字段。
- *json_lines*、*json*：提供结构化解析 JSON 类型日志功能。
- *date*：提供日志内容中有关日期、时间字段相关的解析、转换功能。
- *multiline*：可自定义更为复杂的多行日志类型。
- *kv*：提供结构化解析 Key-Value 类型日志格式功能。

6.2.5 Logstash 错误处理

配置Logstash采集日志数据后，如果在日志采集过程中遇到错误，可以按照报错类型选择对应处理方式。

通过Logstash收集日志数据时，如遇到以下收集错误，请按照对应建议进行处理。

- 日志服务查看到数据乱码

  Logstash 默认支持 UTF8 格式文件编码，请确认输入文件编码是否正确。

- 控制台提示错误

  控制台提示错误`io/console not supported; tty will not be manipulated`时，不影响产品功能，请忽略。

  其它错误类型建议建议参考Google或Logstash论坛，查询帮助信息。

6.3 SDK采集

6.3.1 Producer Library

LogHub Producer Library 是针对应用程序高并发写LogHub类库，Producer Library 和*Consumer Library*是对LogHub的读写包装，降低数据收集与消费的门槛。

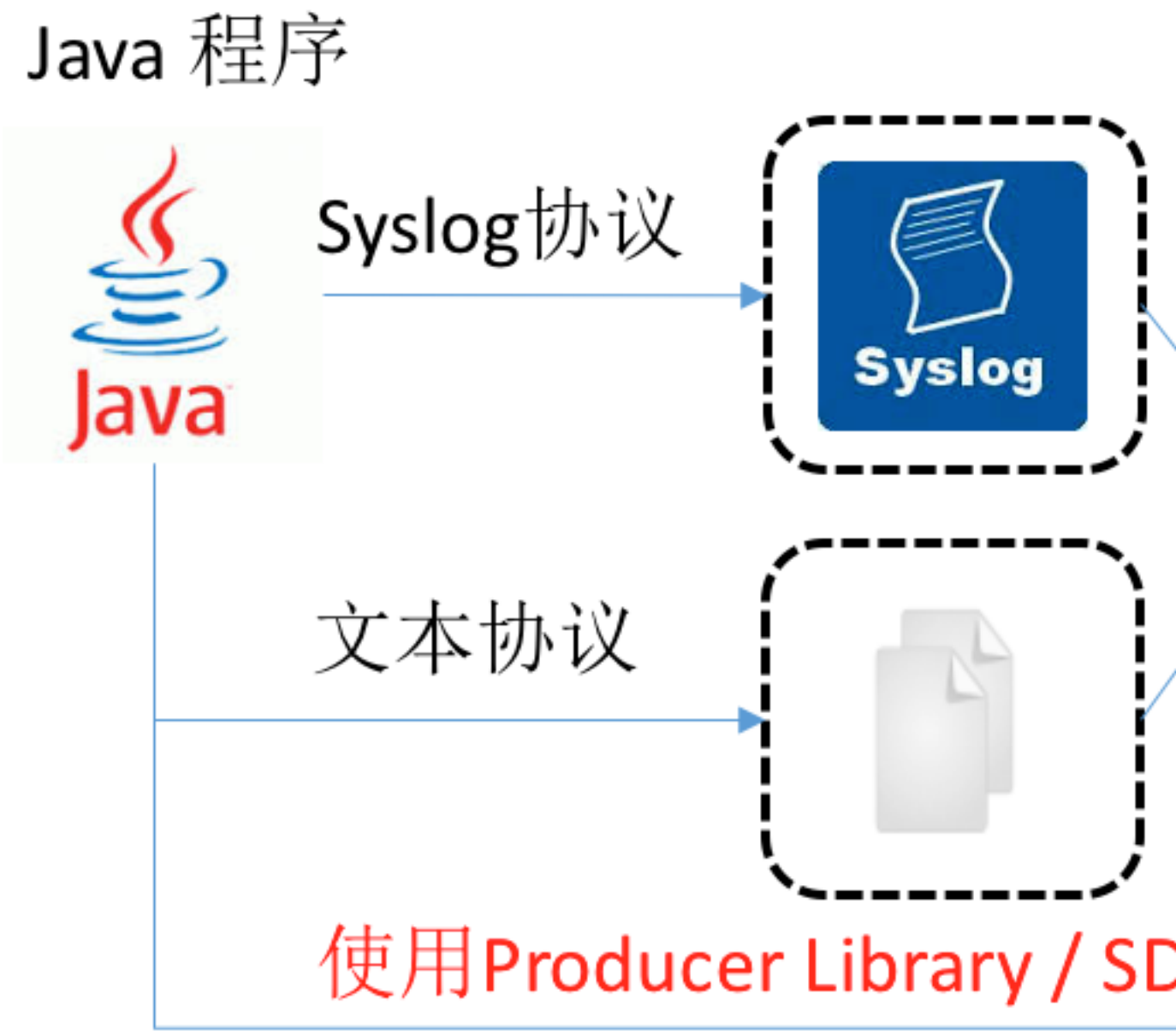
功能特点

- 提供异步的发送接口，线程安全。
- 可以添加多个Project的配置。
- 可以配置用于发送的网络 I/O 线程数量。
- 可以配置merge成的包的日志数量以及大小。
- 内存使用可控，当内存使用达到用户配置的阈值时，Producer 的 send 接口会阻塞，直到有空闲的内存可用。

功能优势

- 客户端日志不落盘：既数据产生后直接通过网络发往服务端。
- 客户端高并发写入：例如一秒钟会有百次以上写操作。
- 客户端计算与 IO 逻辑分离：打日志不影响计算耗时。

在以上场景中，Producer Library 会简化您程序开发的代价，帮助您批量聚合写请求，通过异步的方式发往 LogHub 服务端。在整个过程中，您可以配置批量聚合的参数，服务端异常处理的逻辑等。



以上各种接入方式的对比：

接入方式	优点/缺点	针对场景
日志落盘 + logtail	日志收集与打印日志解耦，无需修改代码。	常用场景
syslog + logtail	性能较好（80MB/S），日志不落盘，需支持 syslog 协议。	syslog 场景

接入方式	优点/缺点	针对场景
SDK 直发	不落盘，直接发往服务端，需要处理好网络 IO 与程序 IO 之间的切换。	日志不落盘
Producer Library	不落盘，异步合并发送服务端，吞吐量较大。	日志不落盘，客户端 QPS 高

配置步骤

- *Java Producer*
- *Log4J1.XAppender(基于Java Producer)*
- *Log4J2.XAppender(基于Java Producer)*
- *LogBack Appender(基于Java Producer)*
- *C Producer*
- *C Producer Lite*

6.3.2 Log4j Appender

Log4j 是 Apache 的一个开放源代码项目，通过使用 Log4j，您可以控制日志信息输送的目的地是控制台、文件、GUI 组件、甚至是套接口服务器、NT 的事件记录器、UNIX Syslog 守护进程等；您也可以控制每一条日志的输出格式；通过定义每一条日志信息的级别，您能够更加细致地控制日志的生成过程。最令人感兴趣的就是，这些可以通过一个配置文件来灵活地进行配置，而不需要修改应用的代码。

通过Alibaba Cloud Log Log4j Appender，您可以控制日志的输出目的地为阿里云日志服务。下载地址及使用说明请参考*Github*。

6.3.3 C Producer Library

LogHub除了Java版本的Producer Library外，还支持C版本的Producer Library和Producer Lite Library，为您提供跨平台、简洁、高性能、低资源消耗的一站式日志采集方案。

Github项目地址以及更多详细说明参见：

- *C Producer Library#推荐服务端使用#*
- *C Producer Lite Library#推荐IOT、智能设备使用#*

6.4 常见日志格式

6.4.1 Apache 日志

apache日志格式和目录通常在配置文件 */etc/apache2/httpd.conf*中。

日志格式

Apache日志配置文件中默认定义了两种打印格式，分别为combined格式和common格式。您也可以添加自定义配置，按需求配置您的日志的打印格式。

- combined格式：

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}
i\"" combined
```

- common格式：

```
LogFormat "%h %l %u %t \"%r\" %>s %b"
```

- 自定义格式：

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}
i\" %D %f %k %p %q %R %T %I %O" customized
```

声明使用了 combined 这种日志格式和写入的文件名。

```
CustomLog "/var/log/apache2/access_log" combined
```

字段说明

字段格式	键名称	含义
%a	client_addr	客户端IP地址。
%A	local_addr	本地私有IP地址。
%b	response_size_bytes	响应字节大小，空值时可能为"-"。
%B	response_bytes	响应字节大小，空值时为0。
%D	request_time_msec	请求时间，单位为毫秒。
%h	remote_addr	客户端地址。
%H	request_protocol_supple	请求协议。
%l	remote_ident	客户端日志名称，来自identd。
%m	request_method_supple	请求方法。
%p	remote_port	服务器端口号。

%P	child_process	子进程ID。
%q	request_query	查询字符串，如果不存在则为空字符串。
"%r"	request	请求内容，包括方法名、地址和http协议。
%s	status	响应的http状态码。
%>s	status	响应的http状态码的最终结果。
%f	filename	请求的文件名。
%k	keep_alive	keep-alive请求数。
%R	response_handler	服务端的处理程序类型。
%t	time_local	服务器时间。
%T	request_time_sec	请求时间，单位为秒。
%u	remote_user	客户端用户名。
%U	request_uri_supple	请求的URI路径，不带query。
%v	server_name	服务器名称。
%V	server_name_canonical	服务器权威规范名称。
%I	bytes_received	服务器接收得字节数，需要启用mod_logio模块。
%O	bytes_sent	服务器发送得字节数，需要启用mod_logio模块。
"%{User-Agent}i"	http_user_agent	客户端信息。
"%{Rererer}i"	http_referer	来源页。

日志样例

```
192.168.1.2 - - [02/Feb/2016:17:44:13 +0800] "GET /favicon.ico HTTP/1.
1" 404 209 "http://localhost/x1.html" "Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_11_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.
2564.97 Safari/537.36"
```

配置Logtail收集Apache日志

1. 在**Logstore**列表界面单击**数据接入向导**图表，进入数据接入向导。
2. 选择数据类型。

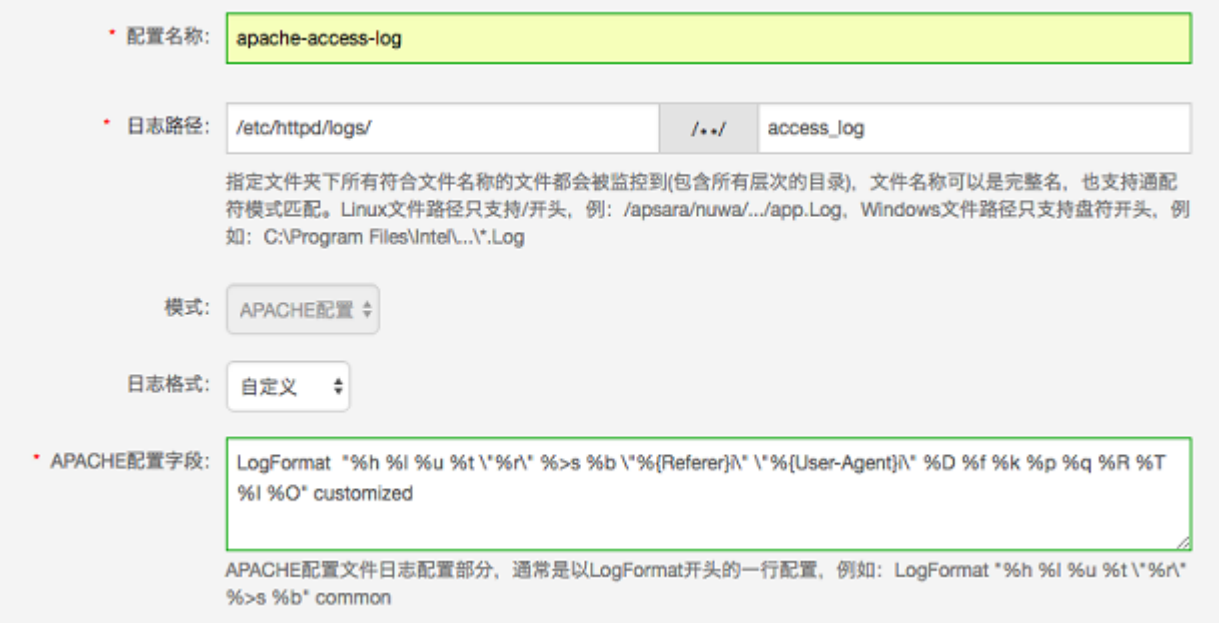
选择**APACHE**访问日志并单击下一步。

3. 配置数据源。

   1. 填写配置名称和日志路径。
   2. 选择日志格式。
   3. 填写**APACHE**配置字段。

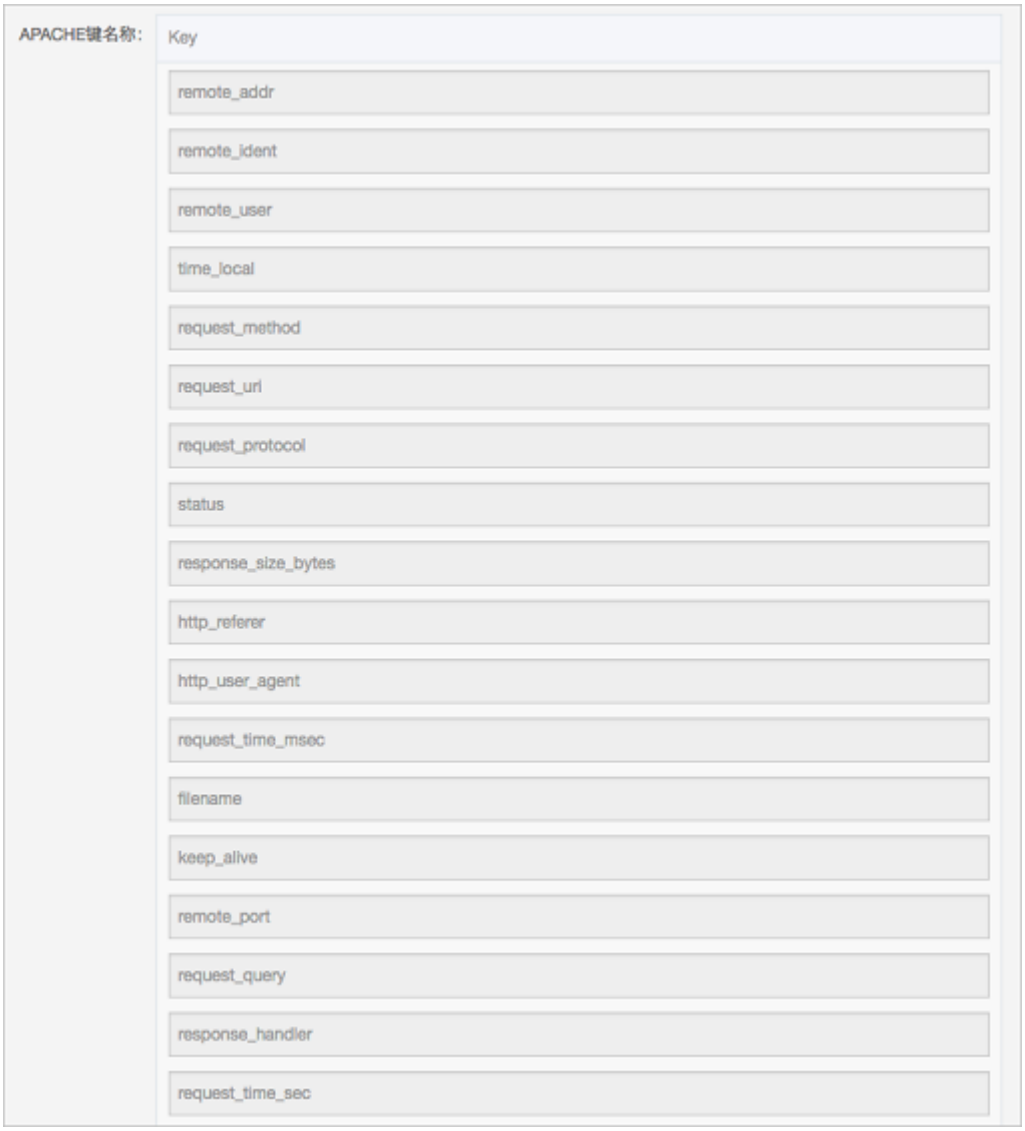
      请填写标准APACHE配置文件日志配置部分，通常以LogFormat开头。

      > 说明：
      > 如您的日志格式为**common**或**combined**格式，此处会自动匹配对应格式的配置字段。

4. 确认**APACHE**键名称。

   日志服务会自动读取您的Apache键。请在当前页面确认APACHE键名称。



5. （可选）配置高级选项。

配置项	详情
本地缓存	请选择是否打开**本地缓存**。当日志服务不可用时，日志可以缓存在机器本地目录，服务恢复后进行续传，默认最大缓存值1GB。
丢弃解析失败日志	请选择解析失败的日志是否上传到日志服务。 开启后，解析失败的日志不上传到日志服务；关闭后，日志解析失败时上传原始日志，其中Key为`__raw_log__`、Value为日志内容。
上传原始日志	请选择是否需要上传原始日志。开启该功能后，原始日志内容会作为__raw__字段与解析过的日志一并上传。
Topic生成方式	• **空-不生成Topic**：默认选项，表示设置Topic为空字符串，在查询日志时不需要输入Topic即可查询。

配置项	详情
	• **机器组Topic属性**：设置Topic生成方式为机器组Topic属性，可以用于明确区分不同前端服务器产生的日志数据。 • **文件路径正则**：选择此项之后，您需要填写下方的**自定义正则**，用正则式从路径里提取一部分内容作为Topic。可以用于区分具体用户或实例产生的日志数据。
自定义正则	如您选择了**文件路径正则**方式生成Topic，需要在此处填写您的自定义正则式。
日志文件编码	• utf8：指定使用UTF-8编码。 • gbk：指定使用GBK编码。
最大监控目录深度	指定从日志源采集日志时，监控目录的最大深度，即最多监控几层日志。最大目录监控深度范围0-1000，0代表只监控本层目录。
超时属性	如果一个日志文件在指定时间内没有任何更新，则认为该文件已超时。您可以对**超时属性**指定以下配置。 • 永不超时：指定持续监控所有日志文件，永不超时。 • 30分钟超时：如日志文件在30分钟内没有更新，则认为已超时，并不再监控该文件。
过滤器配置	日志只有完全符合过滤器中的条件才会被收集。 例如： • **满足条件即收集**：配置`Key:level Regex:WARNING|ERROR`，表示只收集level为WARNING或ERROR类型的日志。 • 过滤不符合条件的数据： — 配置`Key:level Regex:^(?!.*(INFO|DEBUG))`，表示代表不收集level为INFO或DEBUG类型的日志。 — 配置`Key:url Regex:.*^(?!.*(healthcheck)).*`，表示不采集url中带有healthcheck的日志，例如key为url，value为`/inner/healthcheck/jiankong.html`的日志将不会被采集。 更多示例可参考*regex-exclude-word*、*regex-exclude-pattern*。

4. 单击**下一步**。
5. 选择机器组，并单击**应用到机器组**。

   若您未创建机器组，请先单击**+创建机器组**，创建一个机器组。

将Logtail配置应用到机器组之后，日志服务开始按照配置收集Apache日志。您可以在数据接入向导的后续步骤中配置索引、投递日志。

6.4.2 Nginx 日志

nginx 日志格式和目录通常在配置文件 */etc/nginx/nginx.conf*中。

Nginx日志格式

配置文件中定义了Nginx日志的打印格式，即main格式：

```
log_format main  '$remote_addr - $remote_user [$time_local] "$request"
 '
                 '$request_time $request_length '
                 '$status $body_bytes_sent "$http_referer" '
                 '"$http_user_agent"';
```

声明使用了 main 这种日志格式和写入的文件名。

```
access_log /var/logs/nginx/access.log main
```

字段说明

字段名称	含义
remoteaddr	表示客户端IP地址。
remote_user	表示客户端用户名称。
request	表示请求的URL和HTTP协议。
status	表示请求状态。
bodybytessent	表示发送给客户端的字节数，不包括响应头的大小；该变量与Apache模块modlogconfig里的bytes_sent发送给客户端的总字节数相同。
connection	表示连接的序列号。
connection_requests	表示当前通过一个连接获得的请求数量。
msec	表示日志写入的时间。单位为秒，精度是毫秒。
pipe	表示请求是否通过HTTP流水线（pipelined）发送。通过HTTP流水线发送则pipe值为“p”，否则为“.”。
httpreferer	表示从哪个页面链接访问过来的。
“http_user_agent”	表示客户端浏览器相关信息，前后必须加上双引号。

字段名称	含义
requestlength	表示请求的长度。包括请求行，请求头和请求正文。
request_time	表示请求处理时间，单位为秒，精度为毫秒。从读入客户端的第一个字节开始，直到把最后一个字符发送给客户端后进行日志写入为止。
[$time_local]	表示通用日志格式下的本地时间，前后必须加上中括号。

日志样例

```
192.168.1.2 - - [10/Jul/2015:15:51:09 +0800] "GET /ubuntu.iso HTTP/1.0
" 0.000 129 404 168 "-" "Wget/1.11.4 Red Hat modified"
```

配置Logtail收集Nginx日志

1. 在**Logstore**列表界面单击**数据接入向导**图表，进入数据接入向导。
2. 选择数据类型。

   选择**NGINX**访问日志并单击下一步。
3. 配置数据源。
   1. 填写**配置名称**、**日志路径**。
   2. 输入Nginx日志格式。

      请填写标准NGINX配置文件日志配置部分，通常以`log_format`开头。日志服务会自动读取您的Nginx键。
   3. 酌情配置**高级选项**并单击下一步。

      高级选项说明请参考高级选项

   Logtail配置完成后，将此配置应用到机器组即可开始规范收集Nginx日志。

6.4.3 Python日志

Python的 logging 模块提供了通用的日志系统，可以方便第三方模块或者是应用使用。这个模块提供不同的日志级别，并可以采用不同的方式记录日志，比如：文件、HTTP GET/POST、SMTP、Socket等，甚至可以自己实现具体的日志记录方式。logging 模块与 log4j 的机制是一样的，只是具体的实现细节不同。模块提供 logger、handler、filter、formatter。

采集Python日志，推荐您直接使用Logging Handler：

- 使用Log Handler自动上传Python日志
- Log Handler自动解析KV格式的日志
- Log Handler自动解析JSON格式的日志

Python日志格式

日志的格式在 formatter 中指定日志记录输出的具体格式。formatter 的构造方法需要两个参数：消息的格式字符串和日期字符串，这两个参数都是可选的。

Python 日志格式：

```
import logging
import logging.handlers
LOG_FILE = 'tst.log'
handler = logging.handlers.RotatingFileHandler(LOG_FILE, maxBytes =
1024*1024, backupCount = 5) # 实例化 handler
fmt = '%(asctime)s - %(filename)s:%(lineno)s - %(name)s - %(message)s'

formatter = logging.Formatter(fmt)   # 实例化 formatter
handler.setFormatter(formatter)      # 为 handler 添加 formatter
logger = logging.getLogger('tst')    # 获取名为 tst 的 logger
logger.addHandler(handler)           # 为 logger 添加 handler
logger.setLevel(logging.DEBUG)
logger.info('first info message')
logger.debug('first debug message')
```

字段含义

关于 formatter 的配置，采用的是 %(key)s 的形式，就是字典的关键字替换。提供的关键字包括：

格式	含义
%(name)s	生成日志的Logger名称。
%(levelno)s	数字形式的日志级别，包括DEBUG, INFO, WARNING, ERROR和CRITICAL。
%(levelname)s	文本形式的日志级别，包括’DEBUG’、‘INFO’、‘WARNING’、‘ERROR’ 和’CRITICAL’。
%(pathname)s	输出该日志的语句所在源文件的完整路径（如果可用）。
%(filename)s	文件名。
%(module)s	输出该日志的语句所在的模块名。
%(funcName)s	调用日志输出函数的函数名。

格式	含义
%(lineno)d	调用日志输出函数的语句所在的代码行（如果可用）。
%(created)f	日志被创建的时间，UNIX标准时间格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
%(relativeCreated)d	日志被创建时间与日志模块被加载时间的时间差，单位为毫秒。
%(asctime)s	日志创建时间。默认格式是“2003-07-08 16:49:45,896”，逗号后为毫秒数。
%(msecs)d	毫秒级别的日志创建时间。
%(thread)d	线程ID（如果可用）。
%(threadName)s	线程名称（如果可用）。
%(process)d	进程ID（如果可用）。
%(message)s	日志信息。

日志样例

日志输出样例：

```
2015-03-04 23:21:59,682 - log_test.py:16 - tst - first info message
2015-03-04 23:21:59,682 - log_test.py:17 - tst - first debug message
```

常见的Python日志及其正则表达式：

- 日志样例：

```
2016-02-19 11:03:13,410 - test.py:19 - tst - first debug message
```

正则表达式：

```
(\d+-\d+-\d+\s\S+)\s+-\s+([^:]+):(\d+)\s+-\s+(\w+)\s+-\s+(.*)
```

- 日志格式：

```
%(asctime)s - %(filename)s:%(lineno)s - %(levelno)s %(levelname)
s %(pathname)s %(module)s %(funcName)s %(created)f %(thread)d %(
threadName)s %(process)d %(name)s - %(message)s
```

日志样例：

```
2016-02-19 11:06:52,514 - test.py:19 - 10 DEBUG test.py test <module
> 1455851212.514271 139865996687072 MainThread 20193 tst - first
debug message
```

正则表达式：

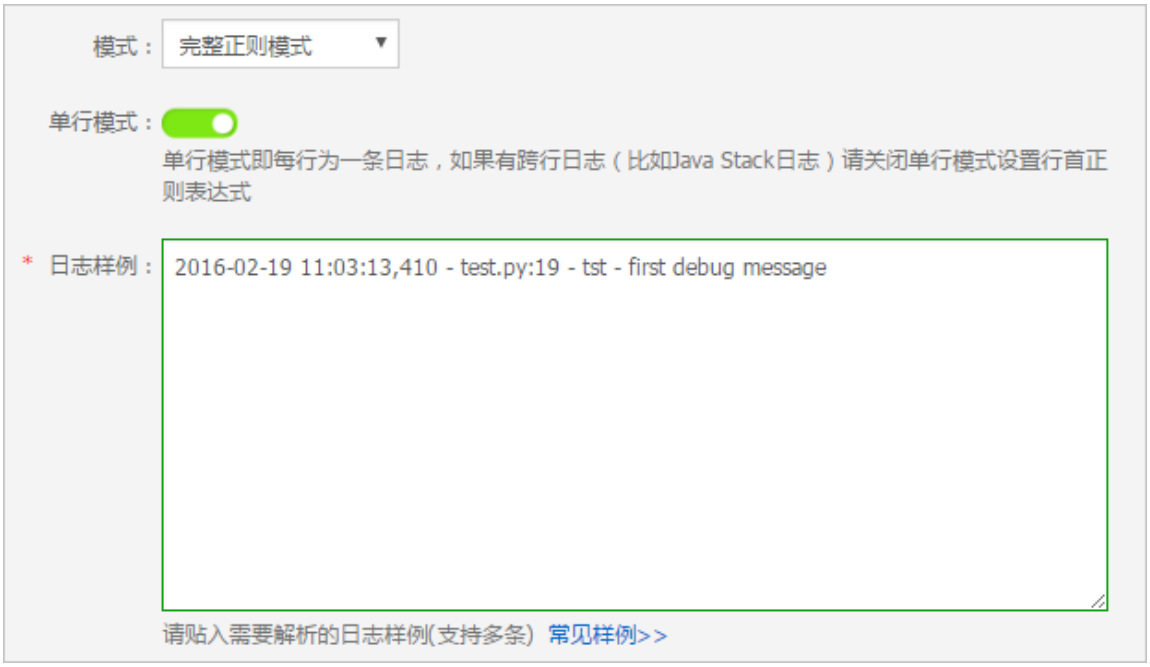
```
(\d+-\d+-\d+\s\S+)\s-\s([^:]+):(\d+)\s+-\s+(\d+)\s+(\w+)\s+(\S+)\s
+(\w+)\s+(\S+)\s+(\S+)\s+(\d+)\s+(\w+)\s+(\d+)\s+(\w+)\s+-\s+(.*)
```

配置Logtail收集Python日志

配置Logtail收集Python日志的详细操作步骤请参考五分钟快速入门，根据您的网络部署和实际情况选择对应配置。

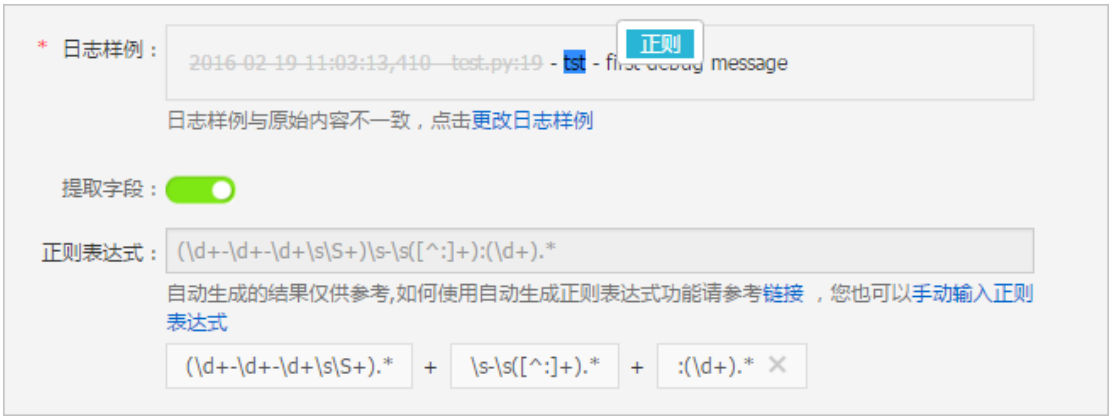
1. 创建Project和Logstore。详细步骤请参考准备流程。
2. 在**Logstore列表**页面单击**数据接入向导**图标，进入数据接入向导。
3. 选择数据类型。

   选择**文本文件**并单击**下一步**。
4. 配置数据源。
   1. 填写**配置名称**、**日志路径**，并选择**日志收集模式**为**完整正则模式**。
   2. 开启**单行模式**。
   3. 输入**日志样例**。



4. 开启**提取字段**。
5. 配置**正则表达式**。
   1. 通过划选生成正则表达式。

      如果自动生成的正则表达式不匹配您的日志样例，可以通过划选的方式生成正则表达式。日志服务支持对日志样例划词自动解析，即对您在划词时选取的字段自动生成正则表达式。请在**日志样例**中划选日志字段内容，并单击**正则**。**正则表达式**一栏中会出现划选内容的正则式。您可以通过多次划选生成日志样例的完整正则表达式。

   2. 调整正则表达式。

      鉴于实际的日志数据格式可能会有细微变动，单击**手动输入正则表达式**，根据实际情况对自动生成的正则表达式做出调整，使其符合收集过程中所有可能出现的日志格式。

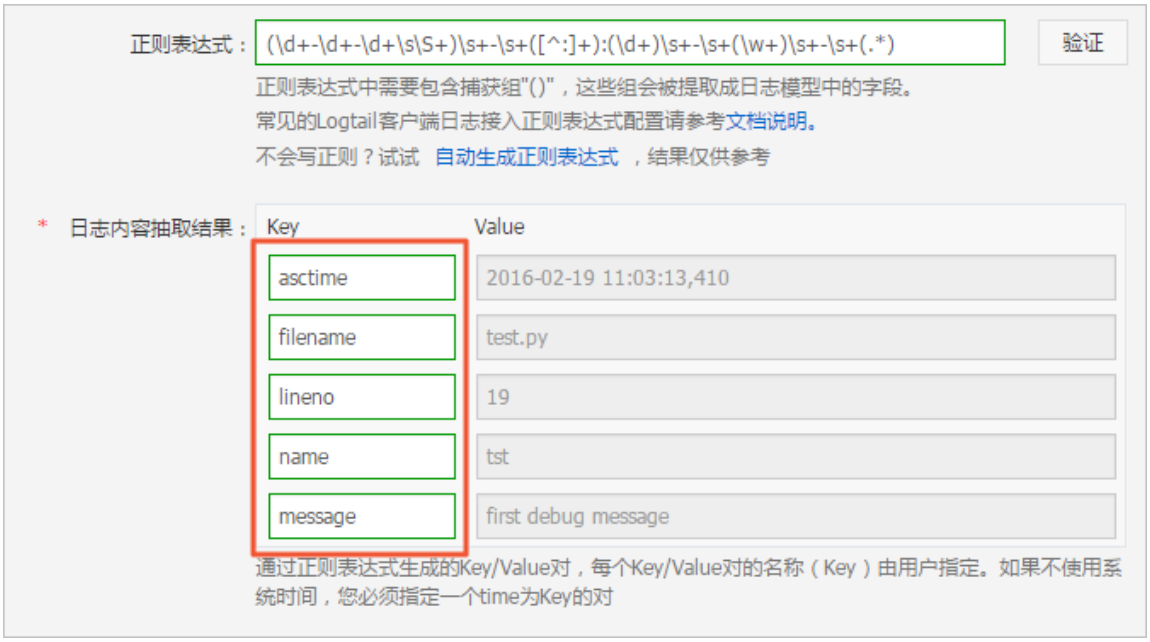
   3. 验证正则表达式。

正则表达式修改完成后，单击验证 。如果正则式没有错误，会出现字段提取的结果，如果有错误请再次调整正则式。

6. 确认日志内容抽取结果。

查看日志字段的解析结果，并为日志内容抽取结果填写对应的Key。

分别为日志字段提取结果取一个有意义的字段名称，比如时间字段的命名为time。如果不使用系统时间，您必须指定Value为时间的字段，并将其Key命名为为time。



7. 开启使用系统时间。

如果使用系统时间，则每条日志时间为Logtail客户端解析该条日志内容的时间。

8. （可选）配置高级选项。

9. 单击 下一步。

Logtail配置完成后，将此配置应用到机器组即可开始规范收集Python日志。

6.4.4 Log4j日志

接入方式

日志服务支持通过以下方式采集log4j日志：

- LogHub Log4j Appender
- Logtail

通过Loghub Log4j Appender采集Log4j日志

详细内容及采集步骤请参考*Log4j Appender*。

通过Logtail采集Log4j日志

Log4j日志分第一代和第二代，本文档以第一代的默认配置为例，讲述如何配置正则式，如果采用Log4j 2，需要修改默认配置，把日期完整打印出来。

```
<Configuration status="WARN">
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss:SSS zzz} [%t] %-
5level %logger{36} - %msg%n"/>
    </Console>
  </Appenders>
  <Loggers>
    <Logger name="com.foo.Bar" level="trace">
      <AppenderRef ref="Console"/>
    </Logger>
    <Root level="error">
      <AppenderRef ref="Console"/>
    </Root>
  </Loggers>
</Configuration>
```

配置Logtail收集Log4j日志的详细操作步骤请参考*Python日志*，根据您的网络部署和实际情况选择对应配置。

在生成正则式的部分，由于自动生成的正则式只参考了日志样例，无法覆盖所有的日志情况，所以需要用户在自动生成之后做一些微调。

Log4j 默认日志格式打印到文件中的日志样例如下：

```
2013-12-25 19:57:06,954 [10.207.37.161] WARN impl.PermanentTairDaoImpl
 - Fail to Read Permanent Tair,key:e:470217319319741_1,result:com
```

```
.example.tair.Result@172e3ebc[rc=code=-1, msg=connection error or
timeout,value=,flag=0]
```

多行日志起始匹配（使用IP信息表示一行开头）：

```
\d+-\d+-\d+\s.*
```

提取日志信息的正则表达式：

```
(\d+-\d+-\d+\s\d+:\d+:\d+,\d+)\s\[([^\]]*)\]\s(\S+)\s+(\S+)\s-\s(.*)
```

时间转换格式：

```
%Y-%m-%d %H:%M:%S
```

样例日志提取结果：

Key	Value
time	2013-12-25 19:57:06,954
ip	10.207.37.161
level	WARN
class	impl.PermanentTairDaoImpl
message	Fail to Read Permanent Tair,key:e:470217319319741_1,result:com.example.tair.Result@172e3ebc[rc=code=-1, msg=connection error or timeout,value=,flag=0]

6.4.5 Node.js日志

Node.js的日志默认打印到控制台，为数据收集和问题调查带来不便。通过log4js可以实现把日志打印到文件、自定义日志格式等功能，便于数据收集和整理。

```
var log4js = require('log4js');
log4js.configure({
  appenders: [
    {
      type: 'file', //文件输出
      filename: 'logs/access.log',
      maxLogSize: 1024,
      backups:3,
      category: 'normal'
    }
  ]
});
var logger = log4js.getLogger('normal');
logger.setLevel('INFO');
logger.info("this is a info msg");
```

```
logger.error("this is a err msg");
```

日志格式

通过log4js实现日志数据存储为文本文件格式后，日志在文件中显示为以下格式：

```
[2016-02-24 17:42:38.946] [INFO] normal - this is a info msg
[2016-02-24 17:42:38.951] [ERROR] normal - this is a err msg
```

log4js分为6个输出级别，从低到高分别为trace、debug、info、warn、error、fatal。

通过**Logtail**收集**Node.js**日志

配置Logtail收集Python日志的详细操作步骤请参考*Python*日志，根据您的网络部署和实际情况选择对应配置。

在生成正则式的部分，由于自动生成的正则式只参考了日志样例，无法覆盖所有的日志情况，所以需要用户在自动生成之后做一些微调。您可以参考以下Node.js日志示例，为您的日志撰写正确、全面的正则表达式。

常见的**Node.js**日志及其正则表达式：

- Node.js日志示例1
  - 日志示例：

    ```
    [2016-02-24 17:42:38.946] [INFO] normal - this is a info msg
    ```

  - 正则表达式：

    ```
    \[([^]]+)]\s\[([^\]]+)]\s(\w+)\s-(.*)
    ```

  - 提取字段：

    `time`、`level`、`loggerName`和`message`。

- Node.js日志示例2：
  - 日志示例：

    ```
    [2016-01-31 12:02:25.844] [INFO] access - 42.120.73.203 - - "GET /
    user/projects/ali_sls_log?ignoreError=true HTTP/1.1" 304 - "http
    ://
    ```

```
aliyun.com/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_3)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.97 Safari/
537.36"
```

- 正则表达式：

```
\[([^]]+)]\s\[(\w+)]\s(\w+)\s-\s(\S+)\s-\s-\s"([^"]+)"\s(\d+)
[^"]+("[^"]+)"\s"([^"]+).*
```

- 提取字段：

  `time`、`level`、`loggerName`、`ip`、`request`、`status`、`referer`和`user_agent`。

6.4.6 wordpress 日志

WordPress 默认日志格式

原始日志样例：

```
172.64.0.2 - - [07/Jan/2016:21:06:39 +0800] "GET /wp-admin/js/password
-strength-meter.min.js?ver=4.4 HTTP/1.0" 200 776 "http://wordpress
.c4a1a0aecdb1943169555231dcc4adfb7.cn-hangzhou.alicontainer.com/wp-
admin/install.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537
.36"
```

多行日志起始匹配（使用 IP 信息表示一行开头）：

```
\d+\.\d+\.\d+\.\d+\s-\s.*
```

提取日志信息的正则表达式：

```
(\S+) - - \[([^\]]*)] "(\S+) ([^"]+)" (\S+) (\S+) "([^"]+)" "([^"]+)"
```

时间转换格式：

```
%d/%b/%Y:%H:%M:%S
```

样例日志提取结果：

Key	Value
ip	10.10.10.1
time	07/Jan/2016:21:06:39 +0800
method	GET
url	/wp-admin/js/password-strength-meter.min.js?ver=4.4 HTTP/1.0
status	200

Key	Value
length	776
ref	http://wordpress.c4a1a0aecdb1943169555231dcc4adfb7.cn-hangzhou.alicontainer.com/wp-admin/install.php
user-agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36

6.4.7 分隔符日志

日志介绍

分隔符日志以换行符作为边界，每一个自然行都是一条日志。每一条日志以固定分隔符连接日志的多个字段，分隔符（Separator）包括制表符、空格、竖线、逗号、分号等单字符。如果字段内部包含分隔符，使用双引号作为引用符（Quote）对字段进行包裹。

常见的分隔符日志有：CSV、TSV 等。

分隔符日志使用分隔符（Separator）将一条日志切分成多个字段，支持**单字符**和**多字符**两种模式。

单字符模式

单字符模式中，您需要指定分隔符，也可以同时指定引用符。

- **分隔符**：日志字段通过单字符的分隔符进行切分，例如制表符（`\t`）、竖线（`|`）、空格、逗号（`,`）、分号（`;`）和不可见字符等单字符。

   **说明：**

  分隔符不支持设置为双引号（`"`）。

  如果双引号作为日志内容，而不是引用符（Quote）出现在日志中，则需要进行转义，日志中应处理为`""`。日志服务解析字段时会自动还原，即将`""`还原为`"`。双引号（`"`）可以作为Quote，在字段的边界单次出现；也可以作为字段内容成对出现（即转义为`""`），其它情况不符合分隔符日志的格式定义，请考虑极简模式、正则模式等其它方式进行字段解析。

  例如，逗号作为分隔符，双引号和逗号作为日志字段中的一部分，需要将包含分逗号的日志字段用Quote包裹，同时将日志字段中的双引号转义为成对的双引号`""`。处理后的日志格式为：`1999,Chevy,"Venture ""Extended Edition, Very Large""","",5000.00`。该日志可以被解析为五个字段：`1999`、`Chevy`、`Venture "Extended Edition, Very Large"`、空字段和`5000.00`。

- 引用符：日志字段内容中包含分隔符时，为避免日志字段被误分割，需要指定引用符（Quote）对内部包含分隔符的日志字段进行包裹隔离。被引用符包裹的内容会被日志服务解析为一个完整字段，字段之间只能存在分隔符。

  > 说明：
  > 如果字段之间包含空格、制表符等非分隔符字符，请修改日志格式。

  引用符可以设置为制表符（`\t`）、竖线（`|`）、空格、逗号（`,`）、分号（`;`）和不可见字符等单字符。

  例如，逗号（`,`）作为分隔符，双引号作为Quote时，日志格式为：`1997,Ford,E350,"ac, abs, moon",3000.00`。该日志可以被解析为5个字段：`1997`、`Ford`、`E350`、`ac, abs, moon`和`3000.00`。其中被Quote包裹的`ac, abs, moon`被看做是一个完整字段。

> 说明：
> 日志服务支持将分隔符和引用符设置为不可见字符。不可见字符是ASCII码中编号为1~31及127的字符，指定分隔符和引用符为不可见字符时，您需要查找不可见字符在ASCII码中对应的十六进制数，输入的格式为`0x`不可见字符在*ASCII*码中对应的十六进制数。例如ASCII码中排行为1的不可见字符填写为`0x01`。



多字符模式

多字符模式中，分隔符可以包括2~3个字符，如（`||`）、(&&&)、（`^_^`）等多字符。多字符分隔符模式下，日志解析完全根据分隔符进行匹配，您无需使用引用符（Quote）对日志进行包裹。



说明：

需确保日志字段内容中不会出现分隔符的完整匹配，否则会产生字段误分割。

例如，分隔符设置为&&的情况下，日志：`1997&&Ford&&E350&&ac&abs&moon&&3000.00`会被解析为5个字段：`1997`、`Ford`、`E350`、`ac&abs&moon`和`3000.00`。

日志示例

- 单字符分隔符日志

```
05/May/2016:13:30:28,10.10.*.*,"POST /PutData?Category=
YunOsAccountOpLog&AccessKeyId=***************&Date=Fri%2C
%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature
=****************************** HTTP/1.1",200,18204,aliyun-sdk-
java
05/May/2016:13:31:23,10.10.*.*,"POST /PutData?Category=
YunOsAccountOpLog&AccessKeyId=***************&Date=Fri%2C
%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature
=****************************** HTTP/1.1",401,23472,aliyun-sdk-
java
```

- 多字符分隔符日志

```
05/May/2016:13:30:28&&10.200.**.**&&POST /PutData?Category=
YunOsAccountOpLog&AccessKeyId=***************&Date=Fri%2C%2028%
20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature=pD12XYLmGxKQ%
2Bmkd6x7hAgQ7b1c%3D HTTP/1.1&&200&&18204&&aliyun-sdk-java
05/May/2016:13:31:23&&10.200.**.**&&POST /PutData?Category
=YunOsAccountOpLog&AccessKeyId=***************&Date=Fri%2C
%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature
=****************************** HTTP/1.1&&401&&23472&&aliyun-sdk-
java
```

配置Logtail收集分隔符日志

配置Logtail收集分隔符日志的详细操作步骤请参考采集文本日志，根据您的网络部署和实际情况选择对应配置。

1. 在**Logstore**列表界面单击**数据接入向导**图表，进入数据接入向导。
2. 选择数据类型。

   选择文本文件并单击下一步。
3. 配置数据源。

   a. 填写配置名称、日志路径，并选择日志收集模式为**分隔符模式**。

   b. 填写日志样例并选择分隔符和引用符。

   请根据您的日志格式选择正确的分隔符与引用符，否则日志数据会解析失败。

说明：

指定分隔符或引用符为不可见字符时，您需要查找不可见字符在ASCII码中对应的十六进制数，输入的格式为0x不可见字符在*ASCII*码中对应的十六进制数。例如ASCII码中排行为1的不可见字符填写为0x01。

图 6-1: 配置数据源



c. 指定日志抽取结果中的Key。

填写日志样例并选择分隔符后，日志服务会按照您选择的分隔符提取日志字段，并将其定义为Value，您需要分别为Value指定对应的Key。

如上日志样例，使用不可见字符0x01进行分割，同时使用不可见字符0x02作为引用符。日志一共包含6个字段，设置Key值分别为：time，ip，url，status，latency，user-agent。

d. 指定日志时间。

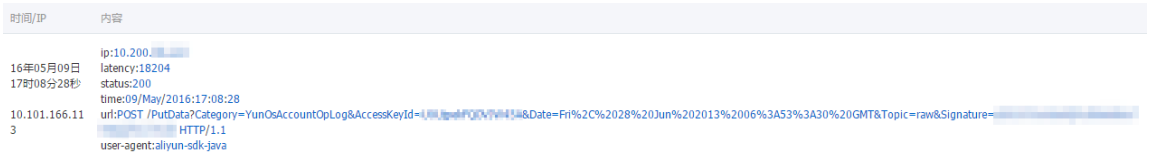
可以选择系统时间作为一条日志的时间，也可以使用日志的一列作为时间，比如选择 time 字段（05/May/2016:13:30:29）作为时间，配置日期格式请参考配置时间格式。

图 **6-2:** 指定日志时间



e. 在控制台上预览日志，确认是否成功收集。

图 **6-3:** 预览日志



6.4.8 ThinkPHP 日志

ThinkPHP 是一个 PHP 语言的 WEB 应用开发框架。

ThinkPHP 日志格式

在ThinkPHP中打印日志按照以下格式：

```
<?php
Think\Log::record('D 方法实例化没找到模型类' );
?>
```

日志示例

```
[ 2016-05-11T21:03:05+08:00 ] 10.10.10.1 /index.php
INFO: [ app_init ] --START--
INFO: Run Behavior\BuildLiteBehavior [ RunTime:0.000014s ]
INFO: [ app_init ] --END-- [ RunTime:0.000091s ]
INFO: [ app_begin ] --START--
INFO: Run Behavior\ReadHtmlCacheBehavior [ RunTime:0.000038s ]
INFO: [ app_begin ] --END-- [ RunTime:0.000076s ]
INFO: [ view_parse ] --START--
INFO: Run Behavior\ParseTemplateBehavior [ RunTime:0.000068s ]
INFO: [ view_parse ] --END-- [ RunTime:0.000104s ]
INFO: [ view_filter ] --START--
INFO: Run Behavior\WriteHtmlCacheBehavior [ RunTime:0.000032s ]
INFO: [ view_filter ] --END-- [ RunTime:0.000062s ]
INFO: [ app_end ] --START--
INFO: Run Behavior\ShowPageTraceBehavior [ RunTime:0.000032s ]
INFO: [ app_end ] --END-- [ RunTime:0.000070s ]
ERR: D 方法实例化没找到模型类
```

配置Logtail收集ThinkPHP日志

配置Logtail收集Python日志的详细操作步骤请参考*Python*日志，根据您的网络部署和实际情况选择对应配置。

在生成正则式的部分，由于自动生成的正则式只参考了日志样例，无法覆盖所有的日志情况，所以需要用户在自动生成之后做一些微调。

由于ThinkPHP是多行日志，而且模式并非固定，可以从日志中提取的字段包括时间、访问IP、访问的URL、以及打印的 Message。其中Message字段包含了多行信息，由于其模式不固定，只能打包到一个字段之中。

ThinkPHP日志的Logtail收集配置参数：

行首正则式

```
\[\s\d+-\d+-\w+:\d+:\d+\+\d+:\d+\s.*
```

正则表达式写成：

```
\[\s(\d+-\d+-\w+:\d+:\d+)[^:]+:\d+\s]\s+(\S+)\s(\S+)\s+(.*)
```

时间表达式写成：

```
%Y-%m-%dT%H:%M:%S
```

6.4.9 Logstash 收集 IIS 日志

使用Logstash采集IIS日志前，需要修改配置文件以解析IIS日志字段。

日志样例

查看 IIS 日志配置，选择格式为 W3C（默认字段设置）保存生效。

```
2016-02-25 01:27:04 112.74.74.124 GET /goods/list/0/1.html - 80 - 66.
249.65.102 Mozilla/5.0+(compatible;+Googlebot/2.1;++http://www.google.
com/bot.html) 404 0 2 703
```

采集配置

```
input {
  file {
    type => "iis_log_1"
    path => ["C:/inetpub/logs/LogFiles/W3SVC1/*.log"]
    start_position => "beginning"
  }
}
filter {
  if [type] == "iis_log_1" {
  #ignore log comments
  if [message] =~ "^#" {
    drop {}
  }
  grok {
    # check that fields match your IIS log settings
    match => ["message", "%{TIMESTAMP_ISO8601:log_timestamp} %{
IPORHOST:site} %{WORD:method} %{URIPATH:page} %{NOTSPACE:querystring}
 %{NUMBER:port} %{NOTSPACE:username} %{IPORHOST:clienthost} %{NOTSPACE
:useragent} %{NUMBER:response} %{NUMBER:subresponse} %{NUMBER:scstatus
} %{NUMBER:time_taken}"]
  }
    date {
    match => [ "log_timestamp", "YYYY-MM-dd HH:mm:ss" ]
      timezone => "Etc/UTC"
  }
  useragent {
    source=> "useragent"
    prefix=> "browser"
  }
```

```
  mutate {
    remove_field => [ "log_timestamp"]
  }
  }
}
output {
  if [type] == "iis_log_1" {
  logservice {
        codec => "json"
        endpoint => "***"
        project => "***"
        logstore => "***"
        topic => ""
        source => ""
        access_key_id => "***"
        access_key_secret => "***"
        max_send_retry => 10
    }
    }
}
```



说明：

- 配置文件格式必须以 UTF-8 无 BOM 格式编码，可以通过notepad++修改文件编码格式。
- *path* 填写文件路径时请使用UNIX模式的分隔符，如：*C:/test/multiline/*.log*，否则无法支持模糊匹配。
- *type* 字段需要统一修改并在该文件内保持一致，如果单台机器存在多个 Logstash 配置文件，需要保证各配置 type 字段唯一，否则会导致数据处理的错乱。

相关插件：*file*、*grok*。

重启Logstash生效

创建配置文件到 *conf* 目录，参考配置 *Logstash* 为 *Windows Service*重启 Logstash 生效。

6.4.10 Logstash 收集 CSV 日志

使用Logstash采集CSV日志前，需要修改配置文件以解析CSV日志字段。采集CSV日志可以使用采集日志的系统时间作为上传日志时间，也可以将日志内容中的时间作为上传日志时间。针对日志时间的不同定义方式，可以通过两种方式配置Logstash收集CSV日志。

使用系统时间作为日志时间上传

- 日志样例

```
10.116.14.201,-,2/25/2016,11:53:17,W3SVC7,2132,200,0,GET,project/
shenzhen-test/logstore/logstash/detail,C:\test\csv\test_csv.log
```

- 采集配置

```
input {
  file {
    type => "csv_log_1"
    path => ["C:/test/csv/*.log"]
    start_position => "beginning"
  }
}
filter {
  if [type] == "csv_log_1" {
  csv {
    separator => ","
    columns => ["ip", "a", "date", "time", "b", "latency", "status",
 "size", "method", "url", "file"]
  }
  }
}
output {
  if [type] == "csv_log_1" {
  logservice {
        codec => "json"
        endpoint => "***"
        project => "***"
        logstore => "***"
        topic => ""
        source => ""
        access_key_id => "***"
        access_key_secret => "***"
        max_send_retry => 10
    }
    }
}
```



说明：

- 配置文件格式必须以 UTF-8 无 BOM 格式编码，可以下载 notepad++ 修改文件编码格式。
- *path* 填写文件路径时请使用 UNIX 模式的分隔符，如：*C:/test/multiline/*.log*，否则无法支持模糊匹配。
- *type* 字段需要统一修改并在该文件内保持一致，如果单台机器存在多个Logstash配置文件，需要保证各配置 *type* 字段唯一，否则会导致数据处理的错乱。

相关插件：*file*、*csv*。

- 重启**Logstash**生效

创建配置文件到 *conf* 目录，参考配置 *Logstash* 为 *Windows Service*重启 Logstash 生效。

使用日志字段内容作为日志时间上传

- 日志样例

```
10.116.14.201,-,Feb 25 2016 14:03:44,W3SVC7,1332,200,0,GET,project/
shenzhen-test/logstore/logstash/detail,C:\test\csv\test_csv_withtime
.log
```

- 采集配置

```
input {
  file {
    type => "csv_log_2"
    path => ["C:/test/csv_withtime/*.log"]
    start_position => "beginning"
  }
}
filter {
  if [type] == "csv_log_2" {
  csv {
    separator => ","
    columns => ["ip", "a", "datetime", "b", "latency", "status", "
size", "method", "url", "file"]
  }
  date {
    match => [ "datetime" , "MMM dd YYYY HH:mm:ss" ]
  }
  }
}
output {
  if [type] == "csv_log_2" {
  logservice {
        codec => "json"
        endpoint => "***"
        project => "***"
        logstore => "***"
        topic => ""
        source => ""
        access_key_id => "***"
        access_key_secret => "***"
        max_send_retry => 10
    }
    }
}
```



说明：

- 配置文件格式必须以 UTF-8 无 BOM 格式编码，可以下载 notepad++ 修改文件编码格式。
- *path* 填写文件路径时请使用 UNIX 模式的分隔符，如：*C:/test/multiline/*.log*，否则无法支持模糊匹配。

- *type* 字段需要统一修改并在该文件内保持一致，如果单台机器存在多个Logstash配置文件，需要保证各配置 *type* 字段唯一，否则会导致数据处理的错乱。

相关插件：*file*、*csv*。

- 重启**Logstash**生效

创建配置文件到 *conf* 目录，参考 配置 *Logstash* 为 *Windows Service* 重启 Logstash 生效。

7 查询与分析

7.1 简介

日志服务提供大规模日志实时查询与分析能力（LogSearch/Analytics），未开启索引时，原始数据可以根据Shard进行类似Kafka的顺序消费；开启索引后，除了支持顺序消费外，还可以对日志数据进行统计与查询。

功能优势

- 实时：写入后可以立即被分析。
- 快速：
  - 查询：一秒内查询（5个条件）可处理10亿级数据。
  - 分析：一秒内分析（5个维度聚合+GroupBy）可聚合亿级别数据。
- 灵活：可以改变任意查询和分析条件，实时获取结果。
- 生态丰富：除控制台提供的报表、仪表盘、快速分析等功能外，还可以与Grafana、DataV、Jaeger等产品无缝对接，并支持Restful API，JDBC等协议。

索引

日志服务中的索引是对日志数据一列或多列的值进行排序的一种结构，使用索引可快速访问日志服务采集到的日志数据。使用日志服务查询与分析功能之前，必须采集到日志数据，并对日志数据开启并配置索引。

日志服务索引包括**全文索引**和**键值索引**。

- **全文索引**：对日志全文内容开启索引，默认的索引会查询日志中所有Key对应的内容，只要有一个命中，就会被查询到。
- **键值索引**：为特定的Key配置索引，配置键值索引后，可以通过查询特定Key的内容，缩小查询范围。

其中，在**键值索引**中，需要指定字段的数据类型。当前，日志服务支持的字段类型包括：*text*、*json*、*long*和*double*。关于索引数据类型的更多信息，请查看索引数据类型简介。

查询方式

- 控制台查询：

在日志服务控制台查询页面指定查询时间段和查询语句进行查询。详细说明和查询语法请参考查询日志和查询语法。

- API查询：

  通过日志服务API中的接口*GetLogs*和*GetHistograms*接口可以查询日志数据。

> 说明：
> 查询日志前，请确认您已采集到日志数据，且已开启并配置索引。

查询分析语句格式

对采集到的日志数据进行实时查询分析时，需要输入查询分析语句（Query）。由查询语句（Search）和分析语句（Analytics）两个部分组成，查询和分析语句之间通过|进行分割。

```
$Search |$Analytics
```

语句类型	是否可选	说明
查询语句（Search）	可选	查询条件，可以包括关键词、模糊、数值、区间范围和组合条件。 如果为空或”*”，表示针对当前时间段所有数据不设置任何过滤条件，即返回所有数据。详细说明请参考查询语法。
分析语句（Analytics）	可选	对查询结果或全量数据进行计算和统计。 如果为空，表示只返回查询结果，不需要做统计分析。详细说明请参考实时分析简介。

注意事项

如果您需要检索的日志数据量很大，例如查询时间跨度非常长，其中数据量在百亿以上时，则一次查询请求无法检索完所有数据。在这种情况下，日志服务会把已有的数据返回给您，并在返回结果中告知您该查询结果并不完整。

同时，服务端会缓存 15 分钟内的查询结果。当查询请求的结果有部分被缓存命中，则服务端会在这次请求中继续扫描未被缓存命中的日志数据。为了减少您合并多次查询结果的工作量，日志服务会把缓存命中的查询结果与本次查询新命中的结果合并返回给您。

因此日志服务可以让您通过以相同参数反复调用该接口来获取最终完整结果。

7.2 实时分析简介

日志服务提供类似于SQL的聚合计算功能，该功能结合了查询功能和SQL的计算功能，对查询结果进行计算。

语法示例：

```
status>200 |select avg(latency),max(latency) ,count(1) as c GROUP BY
method  ORDER BY c DESC  LIMIT 20
```

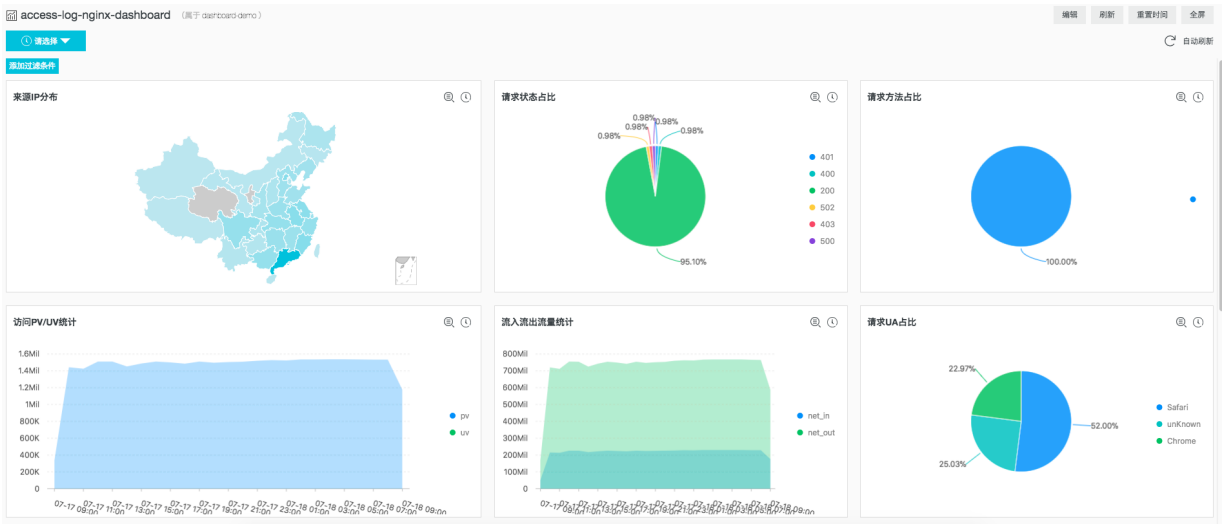
基本语法：

```
[search query] | [sql query]
```

search条件和计算条件以|分割，表示以search query从日志中过滤出需要的日志，并对这些日志进行SQL query计算。search query的语法为日志服务专有语法，参见查询语法。

效果展示

图 7-1: 效果展示



交互分析、仪表盘、Grafana、Datav等更多Demo请单击*DEMO*查看。

前提条件

要使用分析功能，必须在**查询分析设置**中点击SQL涉及的字段下**开启分析**开关，详情请参考简介中设置步骤。

- 如果不开启统计，默认只提供每个shard最多1万行数据的计算功能，而且延时比较高。
- 开启后可以提供秒级别快速分析。
- 开启后只对新数据生效。

- 开启统计后不会产生额外费用。

支持的SQL语法

日志服务支持以下SQL语法，详细内容请点击链接查看。

图 7-2: 支持的SQL语法



- SELECT聚合计算函数：
  - 通用聚合函数
  - 安全检测函数
  - *Map*映射函数
  - 估算函数
  - 数学统计函数

- 数学计算函数
- 字符串函数
- 日期和时间函数
- URL函数
- 正则式函数
- JSON函数
- 类型转换函数
- IP地理函数
- 数组
- 二进制字符串函数
- 位运算
- 同比和环比函数
- 比较函数和运算符
- lambda函数
- 逻辑函数
- 空间几何函数
- 地理函数

- GROUP BY 语法
- 窗口函数
- HAVING语法
- ORDER BY语法
- LIMIT语法
- CASE WHEN和IF分支语法
- unnest语法
- 列的别名
- 嵌套子查询

语法结构

SQL语法结构如下：

- SQL语句中不需要填写from子句和where子句， 默认from表示从当前Logstore的数据中查询，where条件为search query。

- 支持的子句包括SELECT、GROUP BY、ORDER BY [ASC,DESC]、LIMIT、HAVING。

> 说明：
> 默认情况下返回前100个结果，如要返回更多请加上limit n，例如`* | select count(1) as c, ip group by ip order by c desc limit 100`。

内置字段

日志服务内置了一些字段供统计，当用户配置了任何一个有效列后，就会自动加上这些内置字段。

字段名	类型	含义
`__time__`	bigint	日志的时间。
`__source__`	varchar	日志来源IP。在搜索时，该字段是source，在SQL中才会带上前后各两个下划线。
`__topic__`	varchar	日志的Topic。

限制说明

1. 每个Project的最高并发为15。
2. 单列varchar，最大长度为2048，超过后会截断。
3. 默认返回100行数据，不支持翻页。若需要返回更多数据，请使用*LIMIT*语法。

示例

统计每小时的PV、UV和最高延时对应的用户请求，延时最高的10个延时：

```
*|select date_trunc('hour',from_unixtime(__time__)) as time,
     count(1) as pv,
     approx_distinct(userid)  as uv,
     max_by(url,latency) as top_latency_url,
     max(latency,10) as top_10_latency
     group by  1
     order by time
```

7.3 开启并配置索引

使用日志服务查询分析功能之前，请先开启并配置索引。

背景信息

开启并配置索引后，您才可以查询配置索引后采集到的日志数据。请根据您的日志字段内容和查询需求，合理配置索引。



说明：

- 开启查询和统计后意味着数据将会在后台被索引，会产生索引的流量，以及索引对应存储的空间。
- 开启和修改索引后，新的索引配置只对新写入的数据生效。
- 全文索引属性和键值索引属性必须至少启用一种。
- 打开对应字段的统计功能，才能使用SQL进行统计分析。
- 如果配置公网IP、Unix时间戳等*Tag*字段的索引，请配置字段名称为`__tag__:key`，例如`__tag__:__receive_time__`。同时，Tag字段不支持数值类型索引，请将所有Tag字段的索引类型配置为text（文本类型）。例如`__tag__:__receive_time__`字段，在查询时使用模糊查询`__tag__:__receive_time__: 1537928*`或全部匹配字段值`__tag__:__receive_time__: 1537928404`。

采集日志时日志服务会自动将日志来源、时间等信息以Key-Value对的形式添加到日志中，这些字段是日志服务的保留字段。当开启并配置索引时，自动开启这些字段的索引和统计功能。



说明：

`__topic__`和`__source__`的索引分词为空，表示在查询这两个字段时，查询关键字必须完全匹配。

表 7-1: 日志服务保留字段

保留字段名称	说明
`__topic__`	日志主题（Topic）。如果您设置了日志主题，日志服务会自动为您的日志添加日志主题字段，Key为`__topic__`，Value为您的主题内容。
`__source__`	日志来源。该字段表示这条日志的来源设备。
`__time__`	使用SDK写入日志数据时指定的日志时间。

操作步骤

1. 登录日志服务控制台，单击Project名称。
2. 在查询分析列，单击查询。
3. 单击右上角的开启索引。



说明：

若您之前已创建过索引，可以单击**查询分析属性 > 设置**，修改索引。



4. 配置索引。

日志服务支持配置全文索引和键值索引，请至少配置一种。

说明：

若同时配置了全文索引属性和键值索引属性，以键值索引属性设置为准。

索引类型	说明
全文索引	以文本形式对所有的字段进行建索引，Key和Value都是普通文本，都可查询，在查询时不必指定Key的名称。

索引类型	说明
键值索引	配置键值索引后，在查询时要指定Key的名称，当有某个字段配置了键值索引时，该字段上的全文索引不生效。 字段可以配置为多种数据类型，包括： • 文本类型#Text# • JSON类型 • 数值类型#long和double#

a) 配置全文索引。

配置针对全文内容的索引，查询日志时默认查询所有Key对应的内容。

配置	说明	示例
全文索引	对日志全文内容开启索引，默认的索引会查询日志中所有Key对应的内容，只要有一个命中，就会被查询到。	-
大小写敏感	查询时是否区分大小写。其中： • **false** 表示不区分大小写，即查询关键字“INTERNALERROR”和“internalerror”都能查询到对应日志。 • **true**表示区分大小写，只能通过关键字“internalError”查询到对应日志。	-
包含中文	设置是否区分中英文。 • 开启后，如果日志中包含中文，则对中文按照中文语法进行分词，对英文按照分词字符进行分词。 • 关闭后，对所有内容按照分词符进行分词。	-
分词符	根据指定单字符，将日志内容切分成多个关键词。例如一条日志内容为`a,b;c;D-F`。设置分隔符为逗号（,）、分号（;）和连字符（-），可以将日志内容切分为5个关键词”a” “b” “c” “D” “F”。	`, '";=()[]{}? @&<>/:\n\t`

b) 配置键值索引。

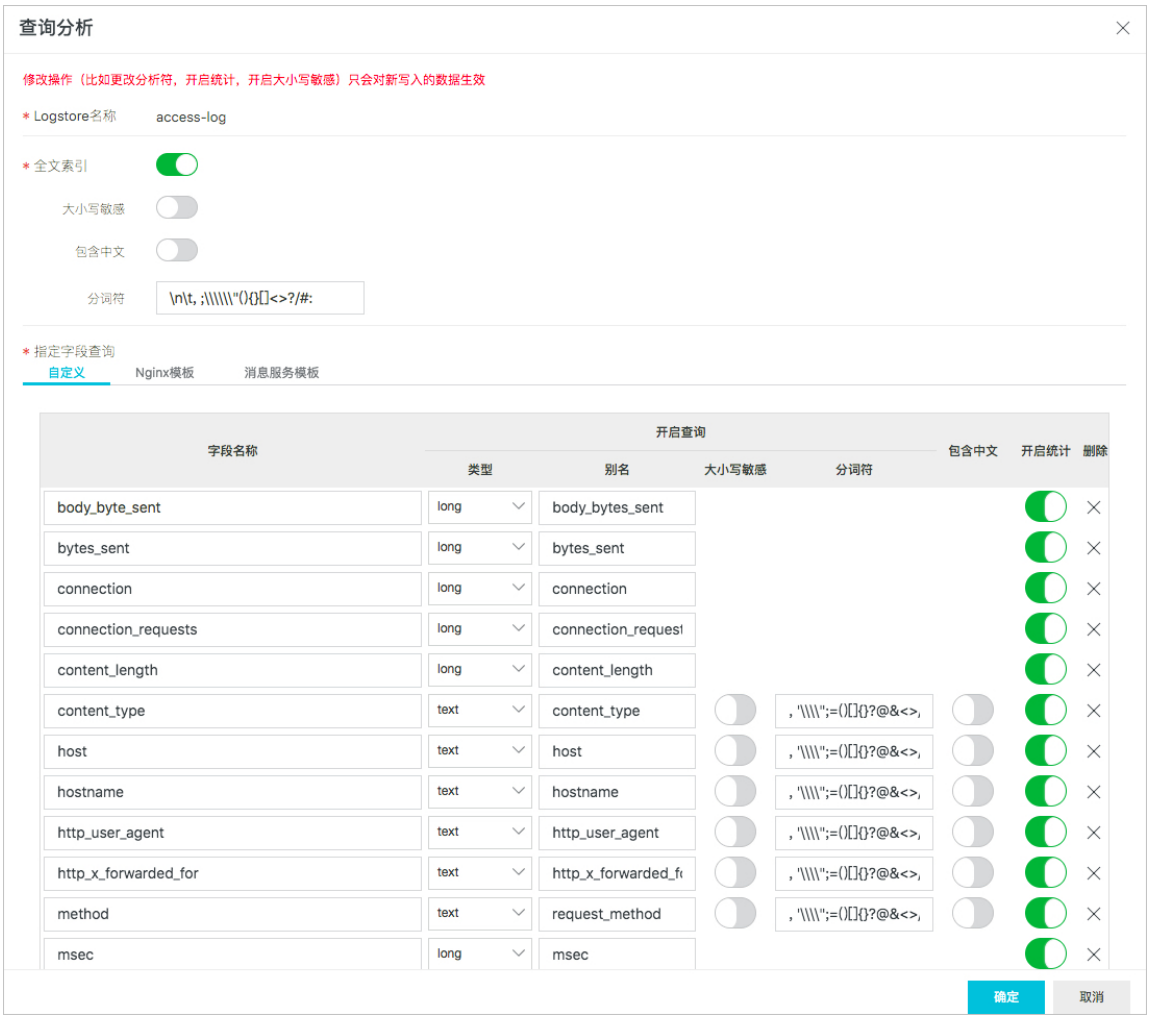
为特定的Key配置索引，配置键值索引后，可以通过查询特定Key的内容，缩小查询范围。

说明：

- 日志服务自动为您创建保留字段__topic__、__source__和__time__的索引和统计功能。
- 本文档以自定义页签为例。Nginx模板和消息服务模板仅用于采集Nginx日志和消息服务日志，不支持自定义配置索引。
- 如果配置公网IP、Unix时间戳等*Tag*字段的索引，请配置字段名称为__tag__:key，例如__tag__:__receive_time__。同时，Tag字段不支持数值类型索引，请将所有Tag字段的索引类型配置为text（文本类型）。例如__tag__:__receive_time__字段，在查询时使用模糊查询__tag__:__receive_time__: 1537928*或全部匹配字段值__tag__:__receive_time__: 1537928404。

配置	说明	示例
字段名称	指定日志字段名称。	_address_
类型	日志字段内容的数据类型，包括： • **text**：指定日志字段内容为文本类型。 • **long**：指定日志字段内容为整数，需要按照数值范围进行查询。 • **double**：指定日志字段内容为小数，需要按照数值范围进行查询。 • **json**：指定日志字段内容为json类型。 说明： 数值类型（long和double类型）不支持设置大小写敏感、分词符和包含中文。	-
别名	列的别名。 别名仅用于SQL统计，在底层存储时，仍然是原始名称，搜索时仍需要使用原始名称。 详细说明请参考列的别名。	address
大小写敏感	查询时，英文字母是否区分大小写。其中， • **false** ：不区分大小写，即查询关键字“INTERNALERROR”和“internalerror”都能查询到样例日志。	-

配置	说明	示例
	• **true**：区分大小写，只能通过关键字“internalError”查询到样例日志。	
分词符	根据指定单字符，将日志内容切分成多个关键词。 例如一条日志内容为`a,b;c;D-F`。设置分隔符为逗号“，”、分号“；”和连字符“-”，可以将日志内容切分为5个关键词”a” “b” “c” “D” “F”。	`, '";=()[]{}?@&<>/:\n\t`
包含中文	开启后，如果日志中包含中文，则对中文按照中文语法进行分词，对英文按照分词字符进行分词；关闭后，对所有内容按照分词符进行分词。	-
开启统计	是否开启统计分析功能。该功能默认开启。开启之后，您可以结合查询语句和分析语句，对日志查询结果进行统计分析。	-



5. 单击确定，结束配置。

> **说明：**
>
> - 索引配置在1分钟之内生效。
> - 开启或修改索引后，新的索引配置只对新写入的数据生效。

7.4 查询日志

开启并配置索引后，可以在查询页面对采集到的日志进行实时查询与分析。

前提条件

- 已采集到日志数据。
- 已开启并配置索引。

操作步骤

1. 登录日志服务控制台，单击Project名称。
2. 在**查询分析**列，单击**查询**。
3. 在搜索框中输入查询分析语句。

   查询分析语句由查询语句和分析语句构成，格式为查询语句|分析语句。详细说明请参考简介。
4. 在页面右上角单击**15分钟（相对）**，设置查询的时间范围。

   您可以选择相对时间、整点时间和自定义时间范围。

   > **说明：**
   > 查询结果相对于指定的时间范围来说，有1min以内的误差。



5. 单击**查询/分析**，查看搜索结果。



日志服务为您提供日志分布直方图、原始日志和统计图表形式的查询分析结果。

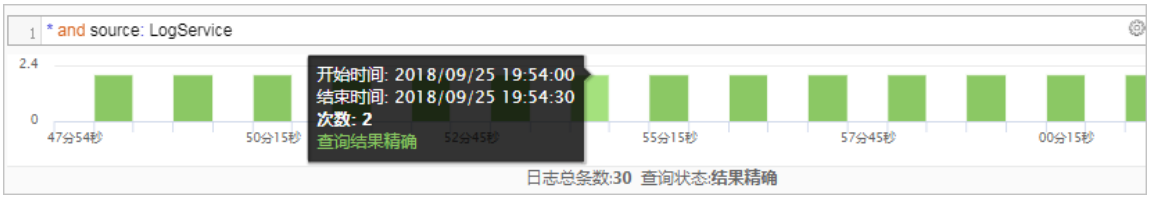
说明：

默认返回100个结果，如果您需要返回更多，请使用*LIMIT*语法。

- **日志分布直方图：**

  日志分布直方图主要展示查询到的日志在时间上分布。

  - 鼠标指向绿色的数据块，可以查看该数据块代表的时间范围和日志命中次数。
  - 单击数据块，可以查看更细时间粒度的日志分布，同时**原始日志**页签中也会同步展示指定时间范围内的日志查询结果。



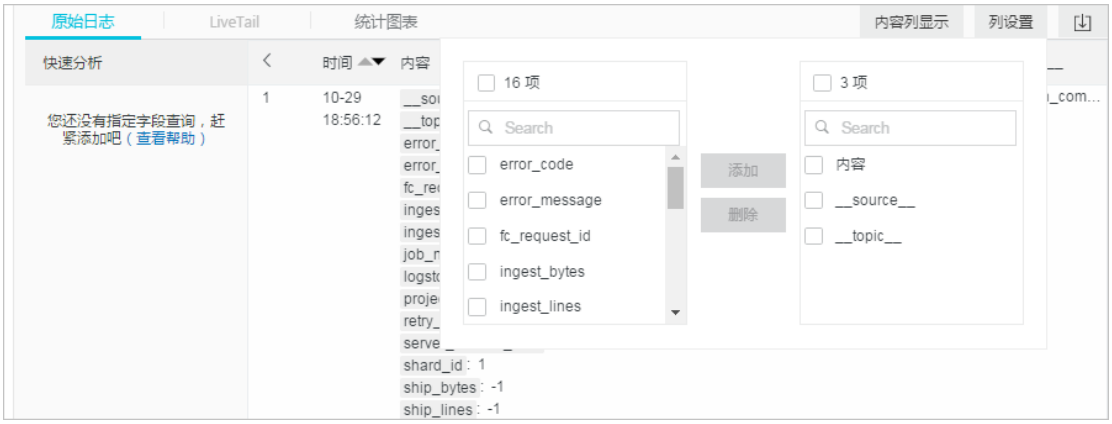
- **原始日志：**

  原始日志页签展示当前查询结果，也就是当前查询条件命中的日志。

  - 快速分析：快速分析功能用于快速分析某一字段在一段时间内的分布情况，详细说明请查看快速分析。
  - 下载日志：单击页签右上角的下载图标，选择下载的范围，并单击**确定**。
  - 设置列：单击页签右上角的**列设置**，勾选字段并单击**添加**，页签中会新增选中字段的列，其中列名称为字段名，内容为每条日志的字段值。

说明：

内容列需要被选中，页签中才会出现日志内容一列。



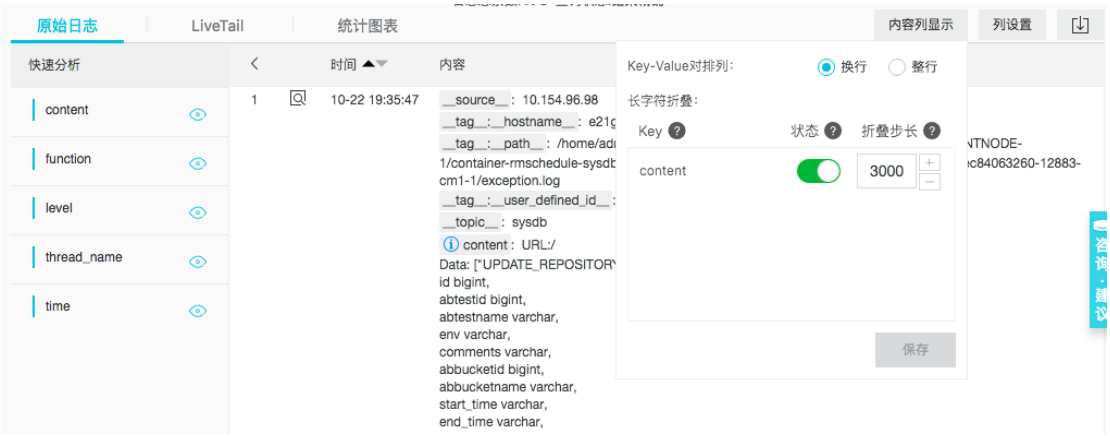
- 设置内容列显示：字段内容如果超出3000字符，会默认折叠处理，并在Key值前显示提醒信息“该字段过长，已做折叠处理”。单击页签右上角的内容列显示，设置**Key-Value**对排列和长字符折叠。

说明：

展开至10000个字符以上时，第10000以外的字符的将做降级处理，降级处理的字符不提供分词的功能。

配置		说明
Key-Value对排列		您可以设置Key-Value对之间为换行显示或整行显示。
长字符折叠	**Key**	当某一Value值超过3000字符时，默认为Value值设置折叠显示，如果日志中不存在过长的Value值，则此处为空。 Key为过长而被折叠的Value对应的Key。
	状态	是否开启Value值折叠。默认为开启状态。 • 开启：表示Key-Value对里的Value值长度超出折叠步长时，会自动对字符进行折叠。单击字符末尾展开按钮可以进行增量展开，增量为折叠步长。 • 关闭：表示超出折叠步长时不折叠。
	折叠步长	Value正常显示的最大长度，也是每次增量展开的长度。

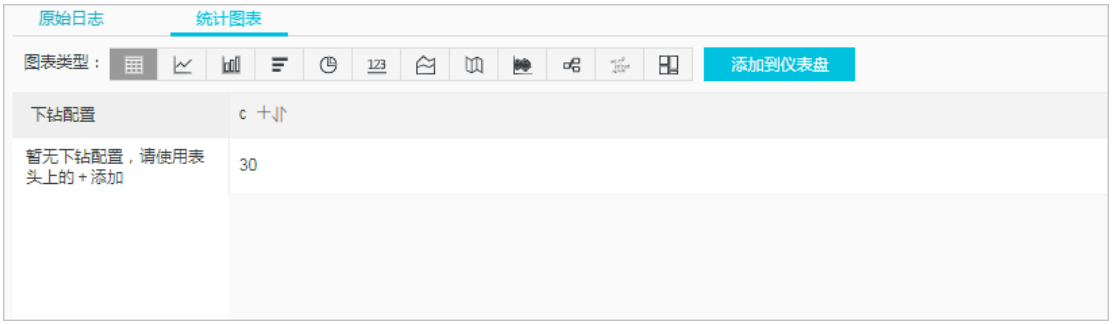
配置		说明
		单位为字符，取值范围为500~10000，默认为3000。



- 统计图表：

  如果在索引设置中开启了统计功能，且在搜索中使用查询分析语句，则可以在**统计图表**页签中查看分析结果。

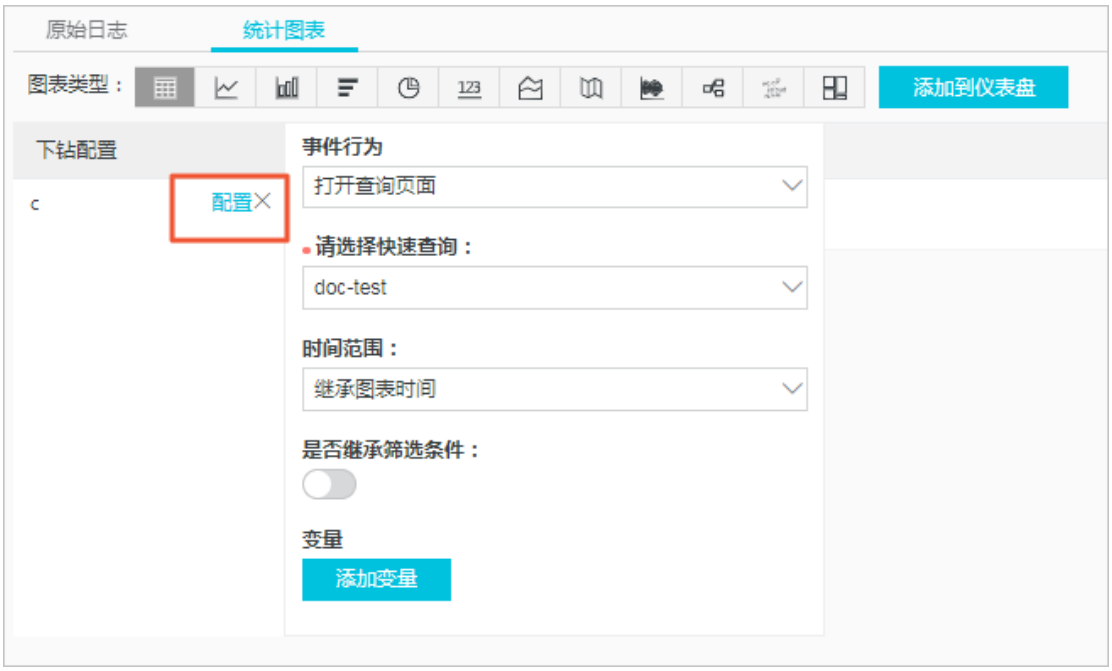
  - 查看分析结果：日志服务为您提供表格、折线图、柱状图等多种类型的统计图表，您可以根据分析需求选用合适的图表类型展示分析结果。



  - 添加图表到仪表盘：仪表盘是日志服务提供的实时数据分析大盘。单击**添加到仪表盘**，将常用的查询语句以图表形式保存到仪表盘中。



- 设置下钻配置：下钻分析是在分析时加深维度，对数据进行层层深入的查看。设置下钻配置并将图表添加到仪表盘，在仪表盘中单击图表值可以获取更深维度的分析结果。详细说明请查看下钻分析。



另外，在查询页面右上角单击**另存为快速查询**和**另存为告警**，可以使用日志服务的快速查询和告警功能。

7.5 索引数据类型

7.5.1 简介

日志服务支持对采集到的日志设置全文或字段索引。设置全文索引后，Value为整条日志；设置字段索引后，每个Key都可以设置数据类型。

数据类型

目前支持的索引数据类型如下：

查询类别	索引数据类型	说明	查询示例
基础查询	*TEXT*	文本类型，可以进行关键词+模糊匹配，支持中文分词。	`uri:"login*" method:"post"`
	Long	数值类型，支持区间查询。	`status>200, status in [200, 500]`
	Double	带浮点数数值类型。	`price>28.95, t in [20.0, 37]`
组合查询	*JSON*	内容为JSON字段，默认为Text类型，支持嵌套模式。可以通过 a.b等路径格式给a层下b元素设置（Text、Long、Double）类型索引，设置后的字段类型以设置为主。	`level0.key>29.95 level0.key2:"action"`
	文本	整条日志当做文本进行查询。	`error and "login fail"`

查询示例

以下一条日志除时间外，还包含4个键值：

序号	Key	类型
0	time	-
1	class	text
2	status	long
3	latency	double
4	message	json

```
0. time:2018-01-01 12:00:00
  1. class:central-log
  2. status:200
  3. latency:68.75
  4. message:
  {
```

```
        "methodName": "getProjectInfo",
        "success": true,
        "remoteAddress": "1.1.1.1:11111",
        "usedTime": 48,
        "param": {
                "projectName": "ali-log-test-project",
                "requestId": "d3f0c96a-51b0-4166-a850-f4175dde7323"
        },
        "result": {
            "message": "successful",
            "code": "200",
            "data": {
                "clusterRegion": "ap-southeast-1",
                "ProjectName": "ali-log-test-project",
                "CreateTime": "2017-06-08 20:22:41"
            },
            "success": true
        }
    }
```

索引设置如下：

图 7-3: 索引设置



其中：

- ①表示可查询json字段中所有string和bool数据。
- ②表示可查询long类型数据。
- ③表示配置的字段可进行SQL分析。

示例：

1. **查询string、bool类型**

   - json内字段无需配置。
   - json map、array自动展开，支持多层嵌套，每一层以”.”进行分割。

```
class : cental*
message.traceInfo.requestId : 92.137_1518139699935_5599
```

```
message.param.projectName : ali-log-test-project
message.success : true
```

2. 查询Double、Long类型

   需要对json内字段独立配置，字段必须不在array。

```
latency>40
message.usedTime > 40
```

3. 组合查询

```
class : cental* and message.usedTime > 40 not message.param.
projectName:ali-log-test-project
```

7.5.2 文本类型

和搜索引擎类似，文本类（Text）数据查询基于词（Term）的命中，因此需要配置分词符、大小写敏感，包含中文（中文分词）选项。

配置说明

大小写敏感

原始日志查询时是否区分大小写。例如原始日志为“internalError”：

- **false**（不区分），即查询关键字“INTERNALERROR”和“internalerror”都能查询到样例日志。
- **true**（区分），只能通过关键字“internalError”查询到样例日志。

分词符

原始日志内容根据分词符可以将日志内容切分成多个关键词。

例如我们要查询如下日志内容：

```
/url/pic/abc.gif
```

- 不设置任何分词符，整个字符串会作为一个独立单词`/url/pic/abc.gif`，只有通过该完整字符串，或通过模糊查询`/url/pic/*`才能找到。
- 如果设置分词符为`/`，则原始日志被切分为`url`、`pic`和`abc.gif`三个单词，可以使用任意一个单词或单词模糊查询都可以找到该日志，例如`url`、`abc.gif`或`pi*`，也可以使用`/url/pic/abc.gif`进行查询(查询时会被拆分为`url and pic and abc.gif`三个条件)。
- 如果设置分词符为`/.`，则原始日志被切分为`url`、`pic`、`abc`和`gif`四个单词。

 说明：

通过设置合理的分词符，可以放宽查询的范围。

包含中文

如果日志中包含中文，需要打开中文分词。例如对如下日志内容：

```
buyer:用户小李飞刀lee
```

默认分词符为":"，则原始日志会被拆分为buyer、用户小李飞刀lee这两个单词，如果搜索用户，则不会返回lee，如果开启**包含中文**选项后，日志服务后台分词器会智能去理解中文含义，并将日志拆分为buyer、用户、小李、飞刀和lee五个单词，无论使用飞刀 或小李飞刀（会被解析为：小李 and 飞刀) 都可以查找到日志。

 **说明：**

中文分词对写入速度会有一定影响，请根据需求谨慎设置。

全文索引

全文查询（索引）默认会将整条日志（除Time以外所有字段、包括Key）作为文本类型，全文查询默认不需要指定key。例如对以下由4个字段组成的日志（时间/status/level/message）。

```
[20180102 12:00:00] 200,error,some thing is error in this field
```

- time:2018-01-02 12:00:00
- level:"error"
- status:200
- message:"some thing is error in this field"

当打开全文索引时，整条日志中会根据Key:Value + "空格"模式组装成一条文本数据，例如：

```
status:200 level:error message:"some thing is error in this field"
```

 **说明：**

- 全文检索时不需要输入前缀，在检索过程中搜索error时（level和message两个字段中error都会被命中）。
- 全文检索需要设置分词符，例如当设置分词符为" "时，可以"status:200"作为一个短语；如果分词符为" :"时，"status"和"200"分别会作为2个独立短语。
- 数值类会被作为文本处理，例如200可以检索到该日志，时间字段（time) 不会被作为文本处理。

- 当输入Key时整条日志也会被命中，例如“status”。

7.5.3 Json类型

Json是由文本、布尔、数值、数组（Array）和图（Map）构成的组合类型数据。

配置说明

文本类数据

对JSON类型字段，会自动识别text/bool类型字段。

例如，以下jsonkey可以通过`jsonkey.key1:"text_value"`、`jsonkey.key2:true`等条件进行查询。

```
jsonkey: {
   key1:text_value,
   key2:true,
   key3:3.14
}
```

数值类数据

非JSON array中的double、long类型数据，可通过设置类型，并指定路径后进行查询。

例如，jsonkey.key3这个字段的类型为double，则查询语句为：

```
 jsonkey.key3 > 3
```

非完全合法JSON

对于非完全合法JSON数据，会尽可能解析有效内容，直到遇到非法部分结束。

例如：

```
"json_string":
{
     "key_1" :  "value_1",
     "key_map" :
      {
             "key_2" : "value_2",
             "key_3" : "valu
```

key_3之后的数据被截断丢失，对于这种缺失的日志可正确解析到`json_string.key_map.key_2`这个字段。

注意事项

- 不支持json object、json array类型。

- 字段不能在json array中。
- bool类型字段可以转成text类型。

查询语法

指定Key查询需要加上JSON中父路径的前缀，文本、数值类查询语法与其他类型相同，详情请参见查询语法。

7.5.4 数值类型

在配置索引时，您可以将字段配置为数值类型，并通过数值范围查询键值。

配置说明

支持类型：`long`（长整数）或者`double`（小数），当设置为数值类型后对于该键的查询只能通过数值范围。

查询示例

查询键值范围为(1000 2000]的longkey，可以使用以下查询方式：

- 数值类查询语法，例如：

```
longKey > 1000 and longKey <= 2000
```

- 也可以使用区间查询语法，例如：

```
longKey in (1000 2000]
```

更多语法请参见查询语法。

7.6 查询语法与功能

7.6.1 查询语法

为了能够帮助您更有效地查询日志，日志服务提供一套查询语法用于设置查询条件。

查询方式

开启并配置索引之后，在日志查询界面输入查询分析语句即可查询日志。

日志查询语句是查询分析语句的前半部分，指定日志查询时的过滤规则，返回符合条件的日志数据。查询语句中支持全文查询和键值查询。

- 全文查询

全文查询时，将整条日志作为一个特殊的Key-Value对，Value为全部的日志内容。全文查询表示在日志内容中查询关键字，即指定查询条件为包含或不包含某个关键字，满足查询条件的日志会作为结果返回。

除了普通的全文查询之外，日志服务还支持短语查询和模糊查询。

- 普通全文查询：指定关键字和规则，包含该关键字并符合规则的日志会作为结果返回。

  例如`a and b`表示查询同时包含关键字`a`和`b`的日志。
- 短语查询：如果需要查询的短语中包含空格，可以将短语用双引号（“”）包裹，表示将双引号中的内容作为一个完整的关键字查询。

  例如`"http error"`表示查询包含关键字`http error`的日志。
- 模糊查询：指定一个64个字符以内的词，在词的中间或者末尾加上模糊查询关键字，即`*`和`?`，日志服务会在所有日志中为您查询到符合条件的100个词，并返回包含这100个词并满足查询条件的所有日志。

  例如`addr?`表示在所有日志中查找以`addr`开头的100个词，并返回包含这些词的日志。

- 键值查询

  为字段都配置字段索引之后，可以指定字段名称和字段内容进行查询。对于double和long类型的字段，可以指定数值范围进行查询。例如设置键值查询语句为`Latency>5000 and Method: Get* and not Status:200`，表示查询`Latency`字段值大于5000、`Method`字段值为`Get`开头，且`Status`字段值不是200的日志。

  根据字段索引中设置的数据类型，您可以进行多种类型的基础查询和组合查询。键值查询示例请参考索引数据类型简介。

注意事项

- 同时配置全文查询和键值查询时，如果索引设置中两者的分词符不同，以键值索引设置为准，使用全文查询方式无法查出有效数据。
- 设置某字段的数据类型为double或long后，才能通过数值范围查询这些字段的数据。若未设置数据类型、或者数值范围查询的语法错误，日志服务会将该查询条件解释成全文索引，可能与您的期望的结果不同。
- 如果将某字段由文本类型改成数值类型，则修改索引之前采集到的数据只支持`=`查询。

运算符

查询语句支持如下运算符：

运算符	说明
and	双目运算符。格式为 `query1 and query2`，表示query1和query2 查询结果的交集。如果多个单词间没有语法关键词，默认是and 的关系。
or	双目运算符。格式为query1 or query2，表示query1和query2 查询结果的并集。
not	双目运算符。格式为query1 not query2 ，表示符合query1 并且不符合query2的结果，相当于query1 - query2。如果只有not query1 ，那么表示从全部日志中选取不包含query1的结果。
(,)	左右括号用于把一个或多个子查询合并成一个查询条件，用于提高括号内查询条件的优先级。
:	用于 key-value 对的查询。term1:term2构成一个 key-value 对。如果 key 或者 value 内有空格、冒号:等保留字符，需要用双引号""把整个 key 或者 value 包括起来。
“	把一个关键词转换成普通的查询字符。左右引号内部的任何一个 term 都会被查询，而不会当成语法关键词。或者在 key-value 查询中把左右引号内的所有 term 当成一个整体。
\	转义符。用于转义引号，转义后的引号表示符号本身，不会当成转义字符，例如"\""。
|	管道运算符，表示前一个计算的基础上进行更多计算，例如 query1 | timeslice 1h | count。
timeslice	时间分片运算符，表示多长时间的数据作为一个整体进行计算，使用方式有 timeslice 1h，timeslice 1m，timeslice 1s 分别表示以 1 小时，1 分钟，1s 作为一个整体。例如 query1 | timeslice 1h | count 表示查询 query 这个条件，并且返回以 1 小时为时间分片的总次数。
count	计数运算符，表示日志条数。
*	模糊查询关键字，用于替代 0 个或多个字符，例如：que*，会返回que 开头的所有命中词。 **说明：** 模糊查询最多返回100个包含符合关键词的日志。

运算符	说明
?	模糊查询关键字，用于替代一个字符，比如qu?ry，会返回以qu开头，以ry结尾，并且中间还有一个字符的所有命中词。
__topic__	查询某个 topic 下数据，可以在 query 中查询 0 个或多个 topic 的数据，例如__topic__:mytopicname。
__tag__	查询某个 tag key 下某个 tag value，例如__tag__:tagkey:tagvalue。
source	查询某个 IP 的数据，例如source:127.0.0.1。
>	查询某个字段下大于某个数值的日志，例如latency > 100。
>=	查询某个字段下大于或等于某个数值的日志，例如latency >= 100。
<	查询某个字段下小于某个数值的日志，例如latency < 100。
<=	查询某个字段下小于或等于某个数值的日志，例如latency <= 100。
=	查询某个字段下等于某个数值的日志，例如latency = 100。
in	查询某个字段处于某个范围内的日志，使用中括号表示闭区间，使用小括号表示开区间，括号中间使用两个数字，数字中间为若干个空格。例如latency in [100 200]或 latency in (100 200]。

说明：

- 运算符不区分大小写。
- 运算符的优先级由高到底排序为:>">( )>and>not>or。
- 日志服务保留以下运算符的使用权，如果您需要使用以下运算符作为查询关键字，请使用双引号包裹起来：sort、asc、desc、group by、avgsum、min、max和limit。

查询示例

查询需求	例句
同时包含 a 和 b 的日志	a and b 或者 a b
包含 a 或者包含 b 的日志	a or b
包含 a 但是不包含 b 的日志	a not b
所有日志中不包含 a 的日志	not a
查询包含 a 而且包含 b，但是不包括 c 的日志	a and b not c
包含 a 或者包含 b，而且一定包含 c 的日志	(a or b) and c

查询需求	例句
包含 a 或者包含 b，但不包括 c 的日志	`(a or b ) not c`
包含 a 而且包含 b，可能包含 c 的日志	`a and b or c`
FILE 字段包含 apsara的日志	`FILE:apsara`
FILE 字段包含 apsara 和 shennong 的日志	`FILE:"apsara shennong"` 或者 `FILE :apsara FILE: shennong` 或者 `FILE: apsara and FILE:shennong`
包含 and 的日志	`and`
FILE 字段包含 apsara 或者 shennong 的日志	`FILE:apsara or FILE:shennong`
file info 字段包含 apsara 的日志	`"file info":apsara`
包括引号的日志	`\"`
查询以 shen 开头的所有日志	`shen*`
查询 FILE 字段下，以 shen 开头的所有日志	`FILE:shen*`
查询 FILE 字段下，值为shen*的所有日志	`FILE: "shen*"`
查询以 shen 开头，以 ong 结尾，中间还有一个字符的日志	`shen?ong`
查询包括以 shen 开头，并且包括以 aps 开头的日志	`shen* and aps*`
查询以 shen 开头的日志的分布，时间片为 20 分钟	`shen*| timeslice 20m | count`
查询 topic1 和 topic2 下的所有数据	`__topic__:topic1 or __topic__ : topic2`
查询 tagkey1 下 tagvalue2 的所有数据	`__tag__ : tagkey1 : tagvalue2`
查询latency大于等于100，并且小于200的所有数据	`latency >=100 and latency < 200`或 `latency in [100 200)`
查询latency 大于100的所有请求	`latency > 100`
查询不包含爬虫的日志，并且http_referer中不包含opx的日志	`not spider not bot not http_referer :opx`
查询cdnIP字段为空的日志	`not cdnIP:""`
查询cdnIP字段不存在的日志	`not cdnIP:*`
查询存在cdnIP字段的日志	`cdnIP:*`

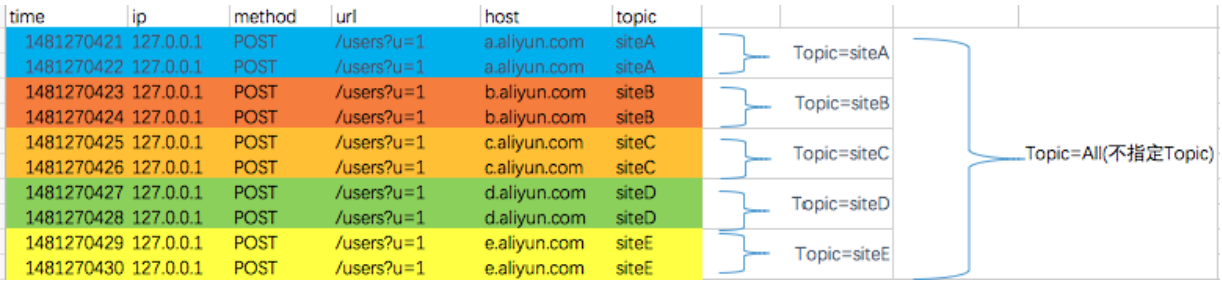
指定或跨 Topic（日志主题）查询

每个 Logstore 根据 Topic 可以划分成一个或多个子空间，当进行查询时，指定 topic 可以限定查询范围，达到更快速度。因此我们推荐对 logstore 有二级分类需求的用户使用 topic 进行划分。

当指定一个或多个 topic 进行查询时，仅从符合条件的 topic 中进行查询。但不输入 topic，默认查询所有 topic 下的数据。

例如，使用Topic来划分不同域名下日志：

图 7-4: 日志Topic



Topic 查询语法：

- 支持查询所有 topic 下的数据，在查询语法和参数中都不指定 topic 意味着查询所有 topic 的数据。
- 支持在 query 中查询 topic，查询语法为 `__topic__:topicName`。同时仍然支持旧的模式，在 url 参数中指定 topic。
- 支持查询多个 topic，例如 `__topic__:topic1 or __topic__:topic2` 表示查询 topic1 和 topic2 下的数据的并集。

模糊查询

日志服务支持单词模糊查询，指定一个64个字符以内的词，在词的中间或者末尾加上模糊查询运算符，即`*`和`?`，日志服务会在所有日志中为您查询到符合条件的100个词，并返回包含这100个词并满足查询条件的所有日志。

限制说明：

- 查询时必须指定前缀，即`*`和`?`不能出现在词的开头。
- 指定的词越精确，查询结果越精确。
- 查询的词超过64个字符，无法使用模糊查询。建议您把查询的词长度缩小到64个字符以下。

7.6.2 LiveTail

LiveTail是日志服务在控制台提供了日志数据实时监控的交互功能，帮助您实时监控日志内容、提取关键日志信息。

背景信息

在线上运维的场景中，往往需要对日志队列中进入的数据进行实时监控，从最新的日志数据中提取出关键的信息进而快速地分析出异常原因。在传统的运维方式中，如果需要对日志文件进行实时监控，需要到服务器上对日志文件执行命令`tail -f`，如果实时监控的日志信息不够直观，可以加上`grep`或者`grep -v`进行关键词过滤。日志服务在控制台提供了日志数据实时监控的交互功能LiveTail，针对线上日志进行实时监控分析，减轻运维压力。

功能优势

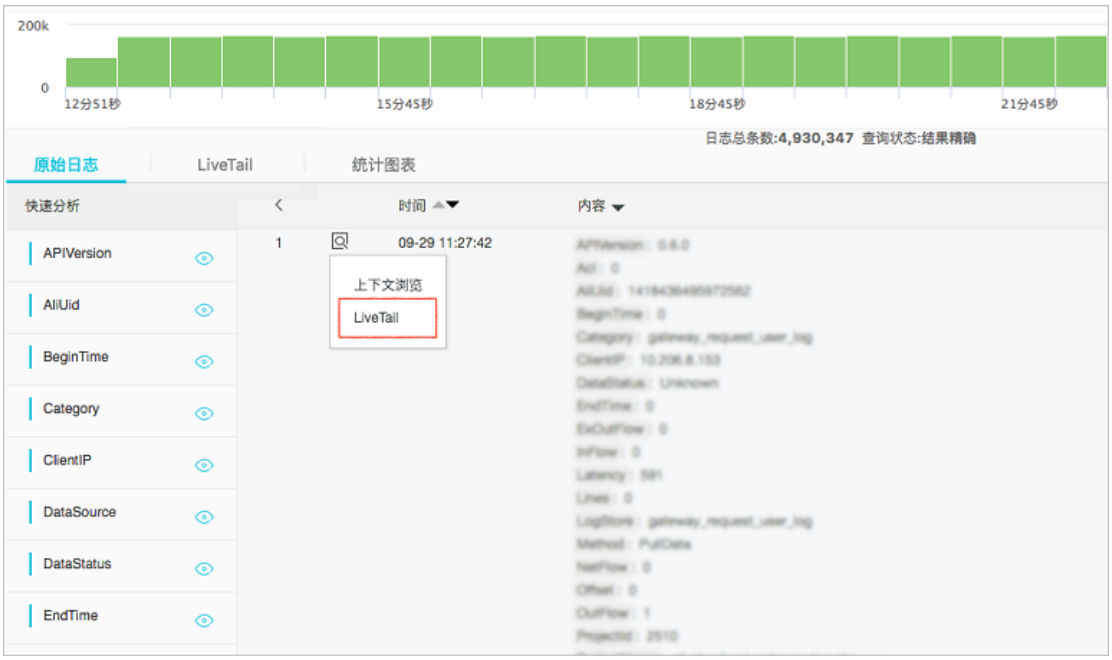
- 监控日志的实时信息，标记并过滤关键词。
- 结合采集配置，对采集的日志进行索引区分。
- 日志字段做分词处理，以便查询包含分词的上下文日志。
- 根据单条日志信息追踪到对应日志文件进行实时监控，无需连接线上机器。

限制说明

- LiveTail功能仅支持Logtail采集到的日志数据。
- 已成功采集到日志数据后，才能使用LiveTail功能。

使用LiveTail实时监控日志

1. 登录日志服务控制台，单击Project名称。
2. 在**查询分析**列下单击**查询**。
3. 您可以通过以下两种方式使用LiveTail功能。
   - 快捷开启LiveTail。
     1. 在**原始日志**页签中，单击指定原始日志的序号右侧图标  ，并选择**LiveTail**。



2. 系统为您自动开启LiveTail，并开始计时。

   其中，**来源类型**、**机器名称**和**文件名称**已预设为指定原始日志的信息。

   开启LiveTail后，Logtail采集到的日志数据会实时显示排列在页面中。最新的日志数据始终在页面底部，且滚动条默认在最下方，即显示最新数据。页面最多显示1000条数据，满1000条后页面自动刷新并重新填充日志数据。



3. （可选）在搜索框中输入关键词。

包含关键词的日志才会显示在监控列表中。筛选包含关键词的日志，便于对特定日志进行实时内容监控。

4. 在实时监控日志的时候，如果某些日志数据可能存在异常需要分析的时候，单击停止**LiveTail**。

   停止LiveTail后，LiveTail计时结束，不再实时更新日志数据。

   针对监控日志时发现的异常，日志服务提供多种分析方式，详细信息请参考使用*LiveTail*分析日志。

- 自定义设置LiveTail。

1. 单击**LiveTail**页签。



2. 配置LiveTail。

配置	是否必选	说明
来源类型	必选	日志的来源，包括： • 普通日志 • Kubernetes • Docker
机器名称	必选	日志来源服务器的名称。
文件名称	必选	日志文件的完整路径及文件名。
过滤条件	可选	关键词，设置关键词后，只有包含关键词的日志才会显示在实时监控窗口中。

3. 单击开启**LiveTail**。

开启LiveTail后，Logtail采集到的日志数据会实时显示排列在页面中。最新的日志数据始终在页面底部，且滚动条默认在最下方，即显示最新数据。页面最多显示1000条数据，满1000条后页面自动刷新并重新填充日志数据。

4. 在实时监控日志的时候，如果某些日志数据可能存在异常需要分析的时候，单击停止**LiveTail**。

   停止LiveTail后，LiveTail计时结束，不再实时更新日志数据。

   针对监控日志时发现的异常，日志服务提供多种分析方式，详细信息请参考使用*LiveTail*分析日志。

使用LiveTail分析日志

停止LiveTail之后，实时监控窗口不再更新显示日志内容，可以对监控过程中发现的问题进行进一步分析排查。

- 查看包含指定字段的日志内容。

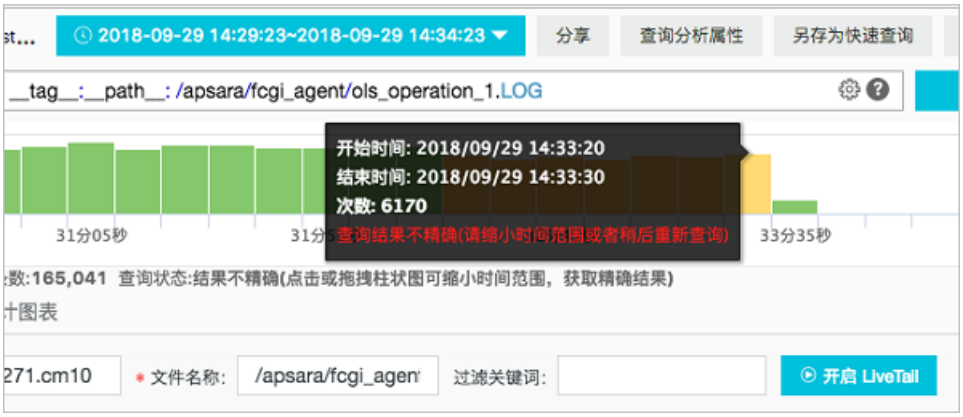
  所有的字段都进行过分词处理，单击指定异常字段内容，页面自动跳转到**原始日志**页签中，按照关键词筛选出该字段相关的所有日志内容。另外也可以对包含该关键字的日志进行上下文查询、查看统计图表等方式进行分析。



- 根据日志分布直方图（histogram）缩小查询的时间范围。

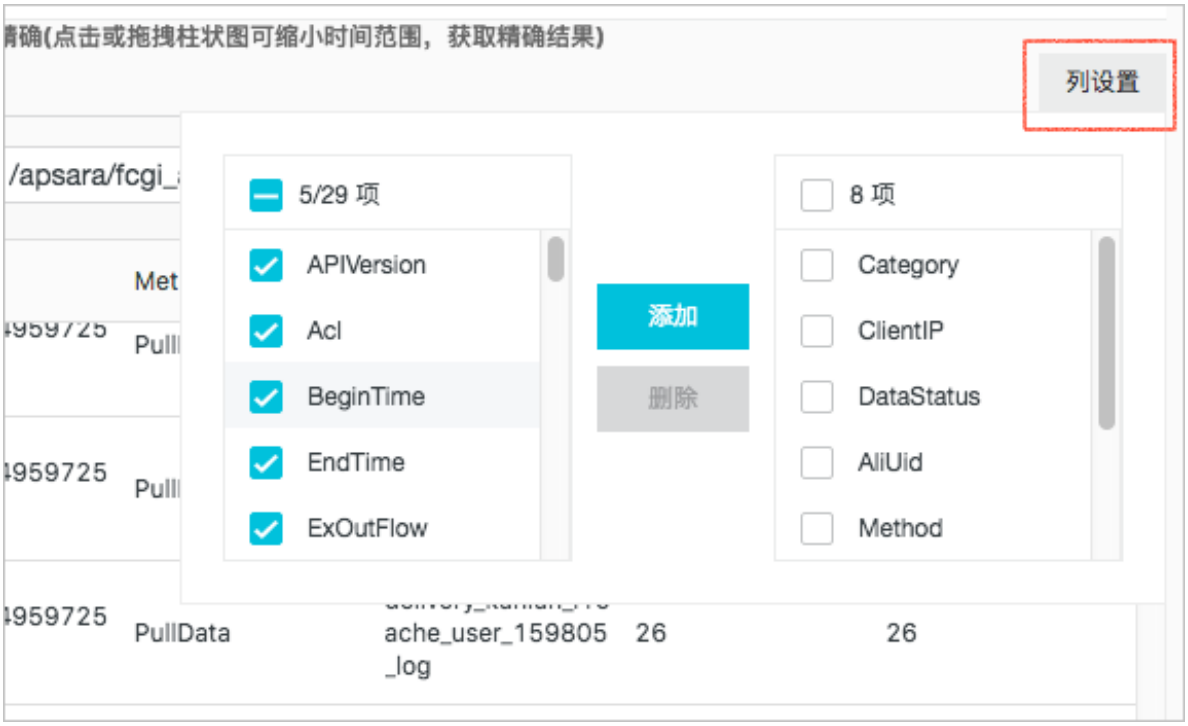
  LiveTail开启时，日志分布直方图也在进行同步更新。如果发现某个时段的日志分布有异常，例如日志数量显著增加时，可以单击该时段的绿色矩形缩小查询的时间范围。跳转后的原始日志内

的时间轴与LivaTail内选择的点击的时间轴是相关联的，可以查看在这段时间内所有的原始日志内容及详细的时间分布。



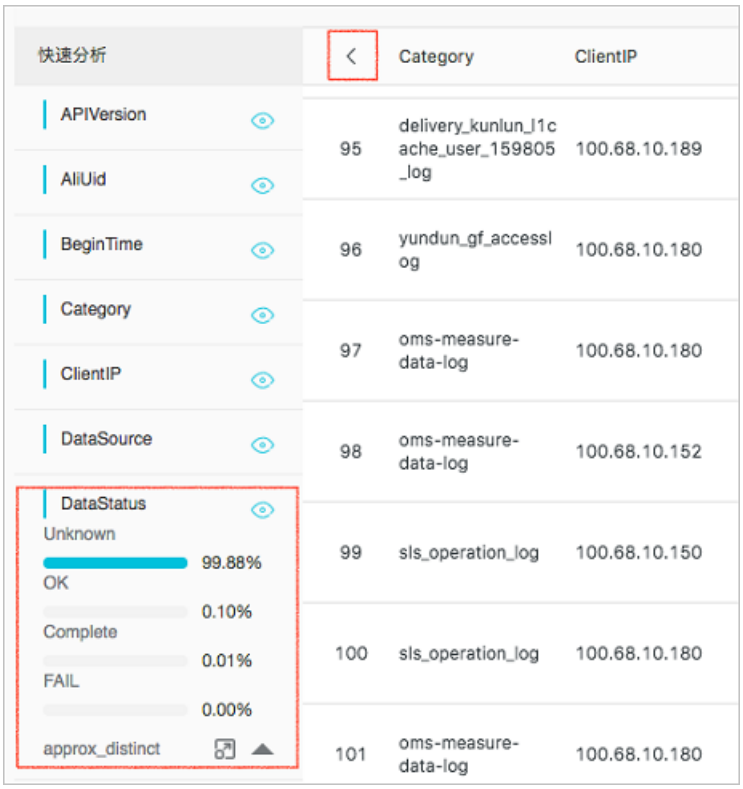
- 通过列设置强调关键信息。

  LiveTail页签中，单击日志列表右上角的**列设置**可以将指定字段单独选设置为一列，使该列的数据更加醒目。可以将需要重点关注的数据设为一列，便于查看和判断异常。



- 对日志数据进行快速分析。

  LiveTail页签中，单击日志列表左上角的箭头，可以展开快速分析区域。快速分析的时间区间是LiveTail开启到停止的时间段，分析的功能与原始日志内提供的快速分析相同。详细说明请参考快速分析。



示例

以下视频为您展示如何使用LiveTail进行实时日志监控及分析。

单击观看

7.6.3 上下文查询

当您展开一份日志文件，每一条日志都记录一个事件，并且往往不是孤立存在的，连续的若干条日志可以回放整个事件序列的发生过程。

日志上下文查询是指定日志来源（机器 + 文件）和其中一条日志，将该日志在原始文件中的前若干条（上文）或后若干条日志（下文）也查找出来，尤其是在 DevOps 场景下对于理清问题来龙去脉来说可谓是一把利器。

日志服务控制台提供专门的查询页面，您可以在控制台查看指定日志在原始文件中的上下文信息，体验类似于在原始日志文件中向上或向下翻页功能。通过查看指定日志的上下文信息，您可以在业务故障排查中快速查找相关故障信息，方便定位问题。

应用场景

例如，O2O 外卖网站在服务器上的程序日志里会记录一次订单成交的轨迹：

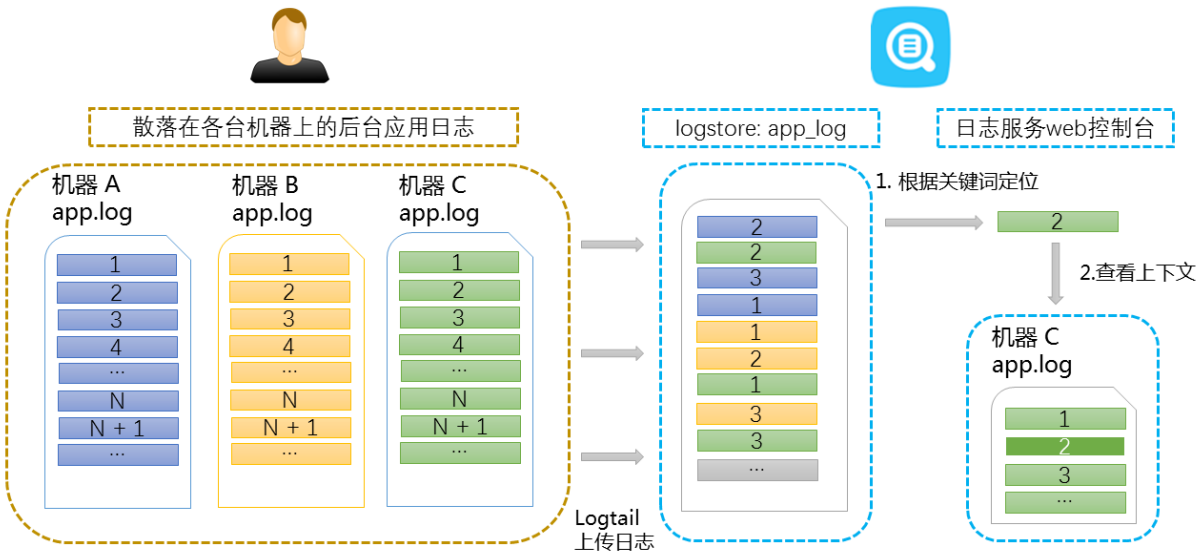
用户登录 > 浏览商品 > 点击物品 > 加入购物车 > 下单 > 订单支付 > 支付扣款 > 生成订单

如果用户下单失败了，运维人员需要快速定位问题原因。传统的上下文查询中，需要管理员相关人员添加机器登录权限，然后调查者依次登录应用所部署的每一台机器，以订单 ID 为关键词搜索应用程序日志文件，帮助判断下单失败原因。

在日志服务中，可以按照以下步骤排查：

1. 到服务器上安装日志采集客户端 Logtail，并到控制台上添加机器组、日志采集配置，然后 Logtail 开始上传增量日志。
2. 到日志服务供控制台日志查询页面，指定时间段根据订单 ID 找到订单失败日志。
3. 以查到的错误日志为基准，向上翻页直到发现与之相关的其它日志信息（例如：信用卡扣款失败）。

图 7-5: 应用场景



功能优势

- 不侵入应用程序，日志文件格式无需改动。
- 在日志服务控制台上可以查看任意机器、文件的指定日志上下文信息，解放了过去需要登录每台机器查看日志文件的痛苦。
- 结合事件发生的时间线索，在日志服务控制台指定时间段快速定位可疑日志后再进行上下文查询，往往可以事半功倍。

- 不用担心服务器存储空间不足或日志文件轮转（rotate）造成的数据丢失，到日志服务控制台上随时可以查看过往的数据。

前提条件

- 使用 Logtail 采集日志 上传数据到日志库，除创建机器组、采集配置以外无需其他配置。
  或使用Producer相关的SDK上传，例如 Producer Library、Log4J、LogBack、C-Producer Library等。
- 开启索引。

> 说明：
> 上下文查询功能暂不支持syslog日志。

操作步骤

1. 进入日志服务控制台。
2. 选择需要的项目，单击项目名称。
3. 在**Logstore**列表页面，选择所需的日志库并单击日志索引列下的**查询**进入查询界面。
4. 输入您的查询分析语句，选择查询时段并单击**搜索**。

   查询结果页中任一条日志的左侧有**上下文浏览**按钮，表明该日志支持上下文查看功能。

   图 **7-6:** 查询日志



5. 选中一条日志，单击**上下文浏览**。在右侧弹出页面中查看目标日志的上下文日志。

6. 使用鼠标在当前页面上下滚动查看选中日志周边的日志信息。如需要继续查看上文和下文，单击 更早 或 更新 进行翻页浏览。

图 7-7: 查询日志



7.6.4 快速查询

快速查询是日志服务提供的一键查询分析功能。

前提条件

已开启并设置索引。

背景信息

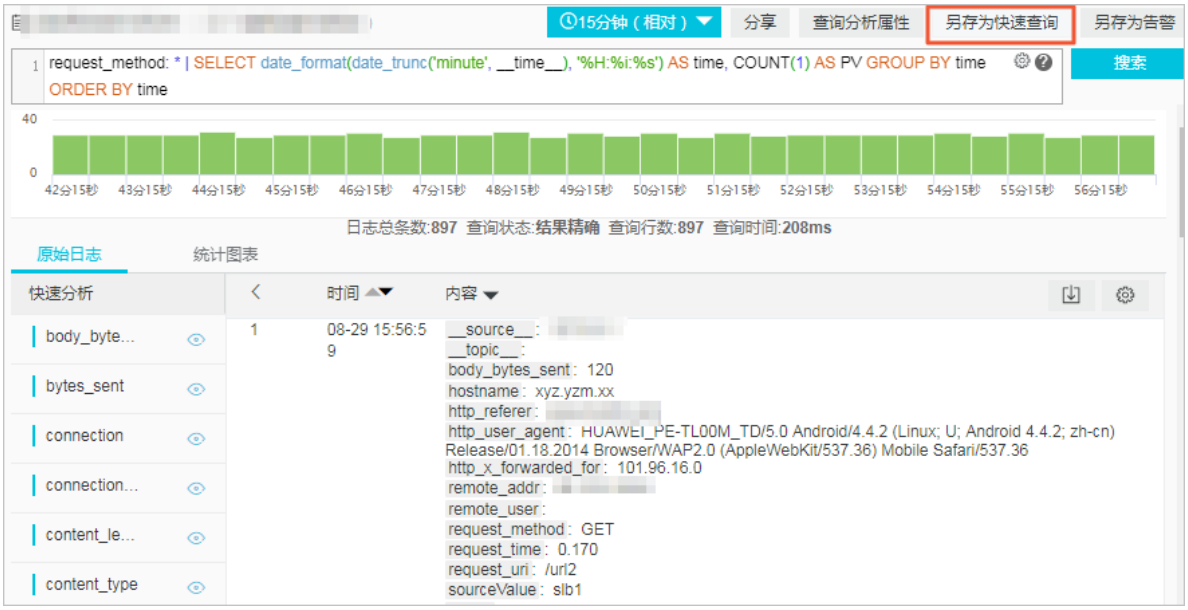
当您需要经常查看某一查询分析语句的结果时，可以将其另存为快速查询，下次进行该查询动作时，不需要手动输入查询语句，只需在查询页面左侧单击该快速查询的名称，即可再次进行该项查

询动作。您也可以在告警规则中使用该快速查询条件。日志服务会定期执行该快速查询语句，并在查询结果满足预设条件时发送告警信息。

设置下钻分析时，如果需要将下钻事件设置**跳转到快速查询**，必须提前配置快速查询，并在查询语句中设置**占位符**。

操作步骤

1. 登录日志服务控制台，单击Project名称。
2. 在**Logstore**列表单击**查询分析**列下的**查询**。
3. 输入您的查询分析语句，设置时间范围，并单击**搜索**。
4. 单击页面右上角的**另存为快速查询**。



5. 设置快速查询属性。
   a) 设置**快速查询名称**。
      - 名称仅支持小写字母、数字、连字符（-）和下划线（_）。
      - 必须以小写字母或数字开头和结尾。
      - 名称长度为3~63个字符。
   b) 确认**日志库**、**日志主题**和**查询语句**。

      若**日志库**和**日志主题**不符合您的需求，请返回至查询页面进入正确的日志库并输入查询语句，再次单击**另存为快速查询**。
   c) （可选）划选查询语句的部分内容，并单击**生成变量**。

生成的变量为占位符变量。您可以在变量名中为占位符变量命名。默认值是您划选时选中的词。

> **说明：**
> 如果其他图表的下钻事件为跳转到这个快速查询，且图表配置的变量和快速查询的变量名相同，单击其他图表时会执行跳转，占位符变量的默认值替换为触发下钻事件的图表值，并以替换变量后的查询语句执行查询。详细信息请查看下钻分析。



6. 单击确定，结束配置。

7.6.5 快速分析

日志服务（Log Service）快速分析功能提供给用户一键交互式查询体验，意图帮助用户快速分析某一字段在一段时间内的分布情况，减少用户索引关键数据的成本。

功能特点

- 支持Text类型字段前100000条数据的前十项分组统计。
- 支持Text类型字段快速生成approx_distinct查询语句。

- 支持long或者double类型字段近似分布直方统计。
- 支持long或者double类型字段快速查找最大项、最小项、平均值或总和。
- 支持将快速分析查询生成查询语句。

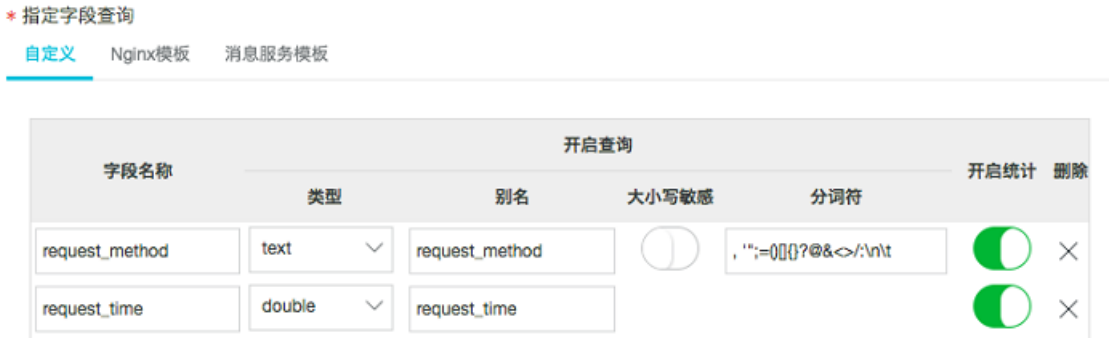
前提条件

快速分析需要用户指定字段查询属性。

1. 指定字段查询需要先开启索引以开启查询分析功能，如何开启索引请参考查询分析。
2. 设置日志中的key为字段名称，并设置类型、别名、分词符等。

   如访问日志中存在request_method和request_time字段，可以参考如下设置。

   图 7-8: 前提条件



使用指南

设置好指定字段查询后，您可以在查询页面的**原始日志**页签左侧**快速分析**一栏处查看到对应的字段。序号顶部按钮可以进行页面折叠，点击**眼睛**按钮即可根据**当前时间区间**、**当前的$Search**条件进行快速分析。

图 7-9: 原始日志



Text类型

- **Text类型分组统计**

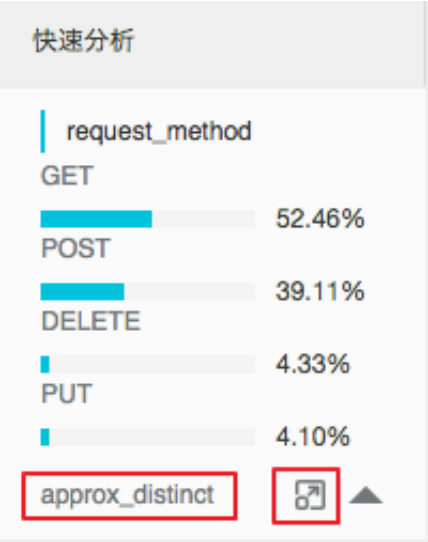
  单击目标字段右侧的眼睛按钮，快速对该**Text**类型字段字段前100,000条数据进行分组，并返回前十项的占比。

  查询语句如下：

```
$Search | select ${keyName} , pv, pv *1.0/sum(pv) over() as
percentage from( select count(1) as pv , "${keyName}" from (select
 "${keyName}" from log limit 100000) group by "${keyName}" order by
pv desc) order by pv desc limit 10
```

  request_method按照分组统计可以得到如下结果，GET请求占大多数：

图 7-10: 分组统计



- 检查字段唯一项的个数

  在快速分析一栏的目的字段下单击**approx_distinct**，即可进行检查操作，即检查`${keyName}`唯一项的个数。

  `request_method`按照分组统计可以得到如下结果，`GET`请求占大多数：

- 将分组统计的查询语句扩展到搜索框

  单击**approx_distinct**右侧的按钮，将分组统计的查询语句扩展到搜索框，便于进一步操作。

long/double类型

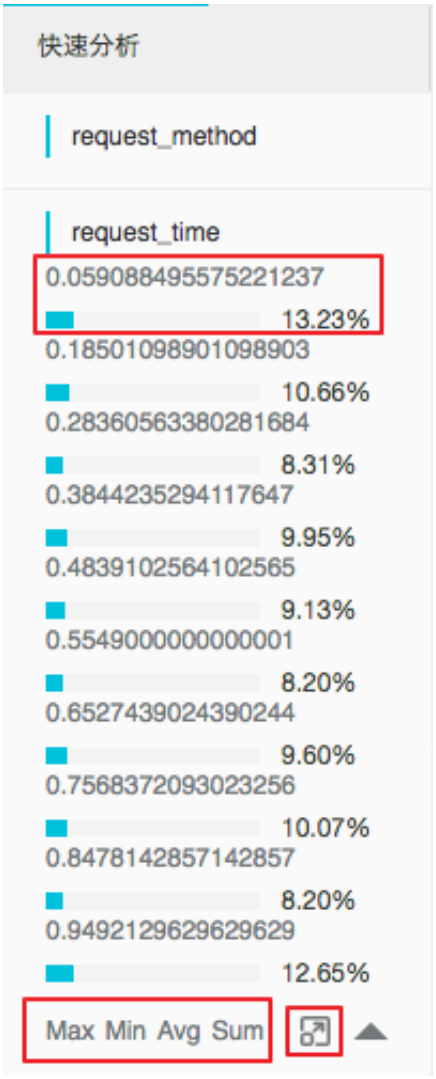
- 近似分布直方统计

  `long/double`类型由于存在多种类型值，计算分组统计意义不大，我们分为10个桶进行近似分布直方统计，查询语句如下：

  ```
  $Search | select numeric_histogram(10, ${keyName})
  ```

  `request_time`按照近似分布直方统计可以得到如下结果，可以知道绝大多数请求时间分布在0.059周围：

图 7-11: 请求分布



- **`MaxMinAvgSum`语句快速分析**

  分别单击目的字段下的`Max`、`Min`、`Avg`、`Sum`，快速查找所有`Max`中的最大项、最小项、平均值和总和。

- **将分组统计的查询语句扩展到搜索框**

  单击`Sum`右侧的按钮，将近似分布直方统计的查询语句扩展到搜索框，便于进一步操作。

7.6.6 其他功能

日志服务查询分析功能除了提供日志内容的各种语句查询能力以外，还提供原始日志、统计图表、上下文查询、快速分析、快速查询、仪表盘、告警等多种扩展功能。

- 原始日志

- 统计图表
- 上下文查询
- 快速分析
- 快速查询
- 仪表盘
- 另存为告警

原始日志

开启索引后，在检索框输入关键字、选择查询时段，单击**搜索**后，即可看到日志数量的直方图、原始日志及统计图表。

日志数量的直方图即日志检索的命中数量在时间上的分布，您可以通过直方图查看某个时间段的日志数量变化，单击长方形区域可以缩小时间范围，查看长方形区域表示的时间范围内的日志命中情况，为您的日志检索结果提供更精细的展示方式。

在原始日志页签中，您可以按时间排序，查看被命中的日志内容。

- 单击列名**时间**旁的三角符号，可切换**时间正序**或**时间倒序**。
- 单击列名**内容**旁的三角符号，可以选择日志内容**换行显示**或者**整行显示**。
- 单击日志内容中的value关键字，可以查看包含该关键字的所有日志内容。
- 单击原始日志页签右上角的**下载**图标，可下载CSV格式的查询结果，单击**设置**图标，可在原始日志显示结果中增加字段为列名的显示列，您可以更直观地在新增列中查看每条原始日志的目标字段内容。
- 单击**上下文**查看该日志的前后各15条日志。更多信息请参考上下文查询。

> **说明：**
> 上下文查询功能目前仅支持使用Logtail上传的数据。

图 7-12: 原始日志



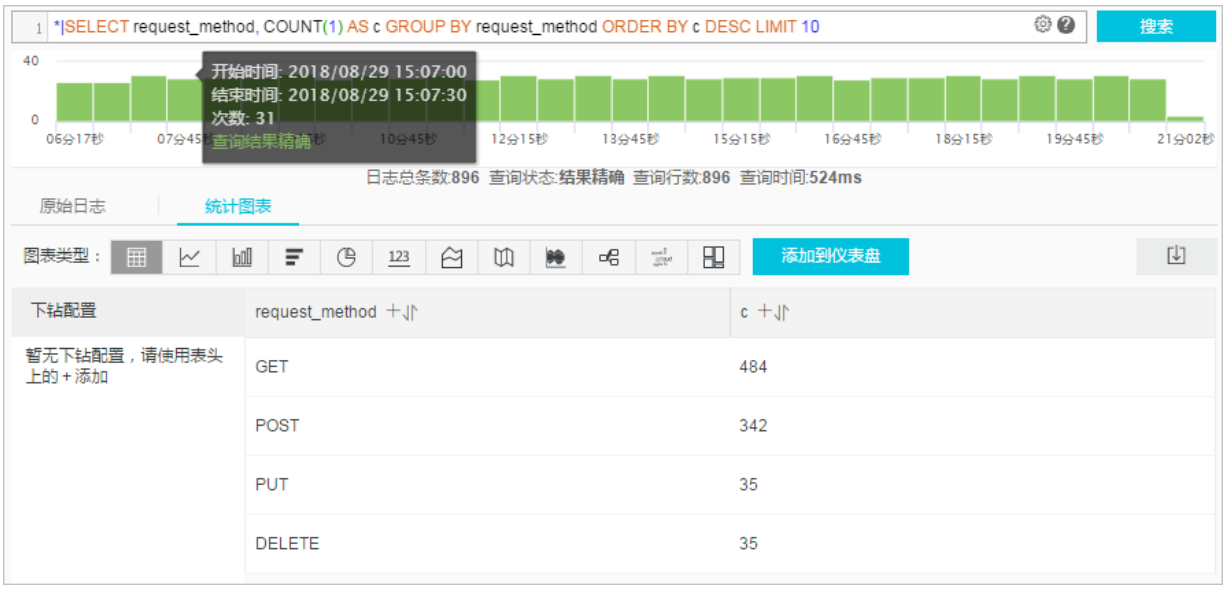
统计图表

开启索引并输入查询和分析语句后，您可以在统计图表页签中查看日志的统计结果。

- 提供表格、折线图等多种统计图表。

  支持根据统计分析的需要选择统计图表类型，支持多种自定义配置。
- 支持将统计图表添加到仪表盘，详细内容请参考仪表盘。
- 支持为统计图表设置下钻分析，改图表添加到仪表盘之后，单及图表数据即触发下钻事件，深化查询维度。

图 7-13: 统计图表



上下文查询

日志服务控制台提供专门的查询页面，您可以在控制台查看指定日志在原始文件中的上下文信息，体验类似于在原始日志文件中向上或向下翻页功能。通过查看指定日志的上下文信息，您可以在业务故障排查中快速查找相关故障信息，方便定位问题。更多信息请参考上下文查询。



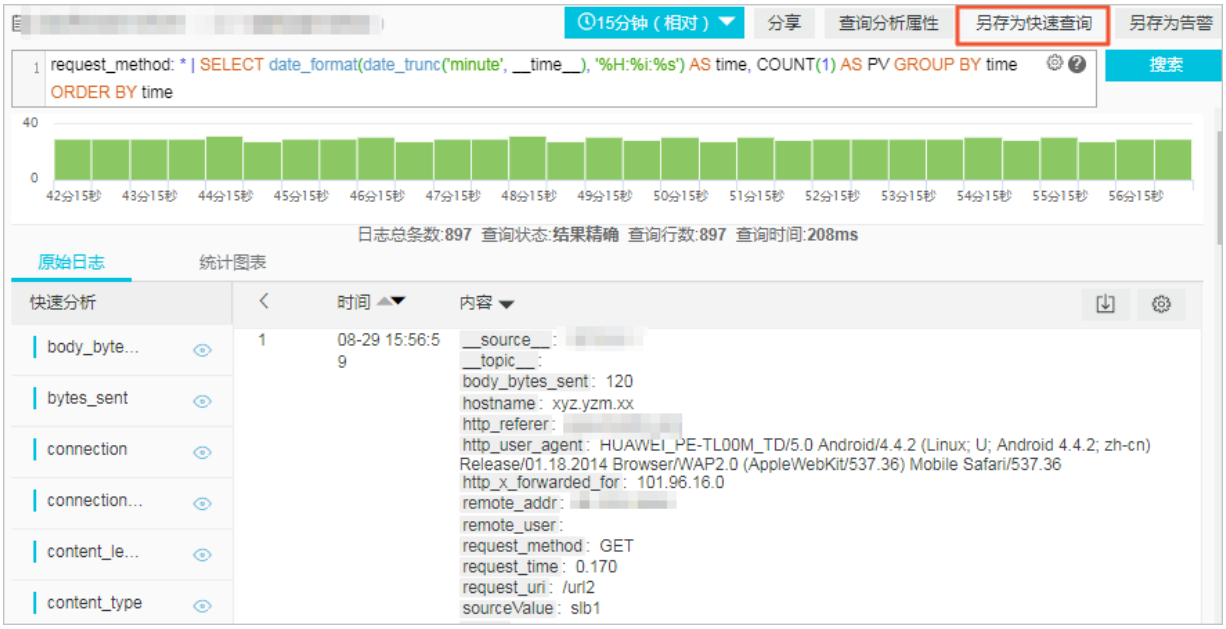
快速分析

日志服务（Log Service）快速分析功能提供给用户一键交互式查询体验，意图帮助用户快速分析某一字段在一段时间内的分布情况，减少用户索引关键数据的成本。详细说明请参考快速分析。



快速查询

日志服务支持将当前的查询动作保存为快速查询，下次进行该查询动作时，不需要手动输入查询语句，只需进入该快速查询页面即可再次进行该项查询动作。详细说明请参考快速查询。

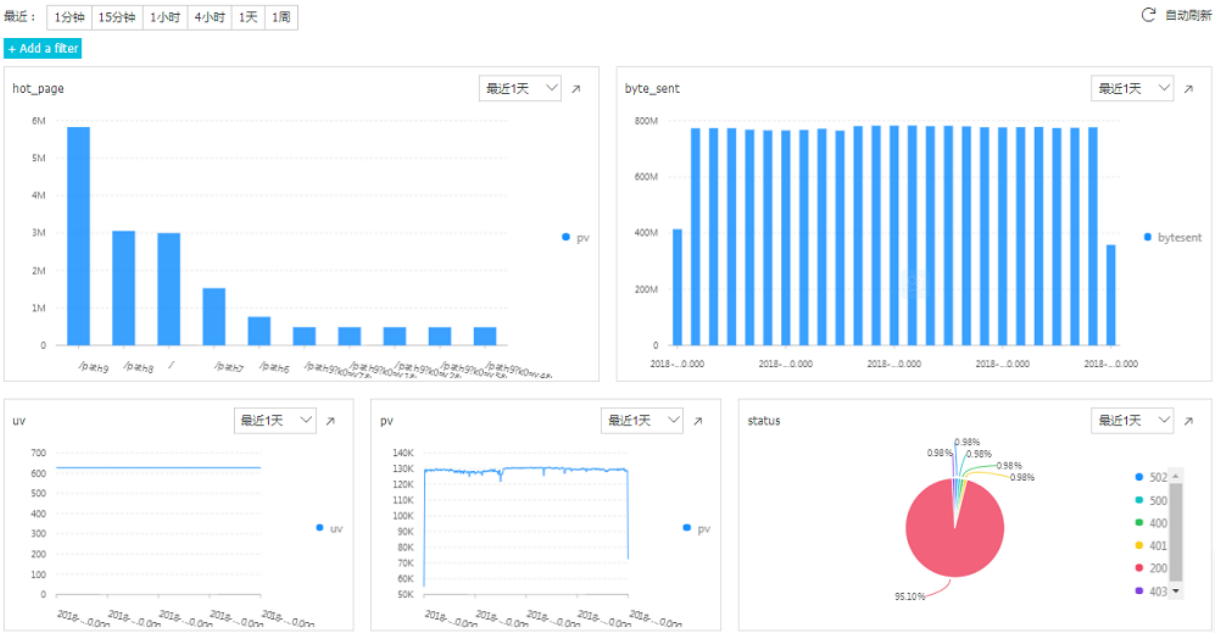


您也可以在告警规则中使用该快速查询条件。设置告警后，日志服务自动定期执行此快速查询，如果查询结果符合预设的阈值，则发送告警信息。

仪表盘

日志服务提供仪表盘功能，支持将查询分析语句进行可视化展示。详细信息请参考仪表盘。

图 7-14: 仪表盘



另存为告警

日志服务支持基于您的日志查询结果进行告警，您可以通过配置规则将具体告警内容以站内通知或者钉钉发送给您。

配置流程为：

1. 配置快速查询。
2. 创建告警规则。
3. 配置通知方式。
4. 查看告警记录。

详细信息请参考设置告警。

7.7 关联外部数据源

7.7.1 简介

应用场景

在日志分析场景中，我们经常会遇到数据分散在各地场景，例如用户操作、行为相关的数据在日志中，同时用户属性、注册信息，资金、道具等状态存在数据库中。类似场景下，需要对用户进行分层统计，将最后的计算结果写入数据库中供报表系统查询。

不同的数据源，存储在不同的存储系统中，如果要实现这种分析，一般的做法是把数据搬迁到一个统一的存储系统中，再进行分析。搬迁数据既涉及到网络传输，又涉及到数据的清洗和格式化归一，耗时又耗精力。导致用户的大部分精力浪费在数据搬迁上。

日志服务提供的外部存储API支持以下功能：

- 通过API为外部存储定义映射，不需要搬迁数据。
- 提供统一的查询分析引擎，支持在一个SQL中对日志、外部存储等多种数据源进行JOIN操作。
- 支持将多种分析结果保存到外部存储中。

功能优势

- 节省精力
  - 节省数据搬迁的人力。不同的存储系统，格式和API都不同，要把数据搬迁到同一个系统中，涉及到复杂的数据转换。使用日志服务外部存储不需要搬迁数据到同一个存储系统中，节约数据搬迁的人力。
  - 节省数据维护的人力。搬迁后，数据一旦有更新，还要经常维护。在日志服务轻量级的运行时联合分析平台中，避免数据搬迁，节省了用户精力，解放用户生产力，可以把有限的开发人力投入到主营业务中。
- 方便快捷
  - 通过SQL分析数据，可在秒级别获得结果。
  - 将常用视图添加到仪表盘，打开仪表盘页面即可快速查看相关信息。

支持的外部存储

表 7-2: 支持的外部存储

外部存储名称	支持从外部数据源读取	支持写入外部数据源	支持的创建方式	支持的Region
MySQL	支持	支持	API、SDK、CLI	华东一（hangzhou）、华北一（qingdao）、华东二（shanghai）、华北二（beijing）
OSS	支持	支持	SQL create table	所有地域

配置方式

- 外部MySQL存储
  - 配置：外部*MySQL*存储
  - 最佳实践：数据库与日志关联分析
- 外部OSS存储
  - 配置：外部*OSS*存储
  - 最佳实践：日志服务与*OSS*关联分析

7.7.2 关联MySQL数据源

日志服务查询分析引擎，提供跨LogStore和ExternalStore的查询分析功能，使用SQL的join语法把日志和用户元信息关联起来。用户可以用来分析跟用户属性相关的指标。

除在查询过程中引用ExternalStore之外，日志服务还支持将计算结果直接写入ExternalStore中（例如MySQL），方便结果的进一步处理。

创建外部MySQL存储的最佳实践：数据库与日志关联分析。

配置方式

- 外部存储API

  API支持创建、更新、删除、列表等接口，具体API请参考API文档。
- 外部存储管理工具

  外部存储的操作可以通过Python SDK或者CLI（命令行工具）完成。CLI请参考*CLI*文档。

操作步骤

1. 采集日志到日志服务。

   请参考采集方式，选择对应方式采集日志到日志服务。
2. 连接到MSQL数据库，并创建MySQL表。
3. 添加白名单。

   请参考设置白名单设置RDS白名单：`100.104.0.0/16`。

   如果是普通MySQL数据库，请添加该地址到安全组。
4. 创建ExternalStore。

   您可以选择多种方式创建ExternalStore，此处以Python CLI方式为例。

1. 执行以下命令安装CLI。

```
pip install -U aliyun-log-cli
```

2. 创建ExternalStore，指定所属的Project，以及ExternalStore的配置文件*/root/config.json*。

```
aliyunlog log create_external_store --project_name="log-rds-demo"
 --config="file:///root/config.json"
```

在配置文件*/root/config.json*中，指定外部存储的名称，以及外部存储的参数等信息。



说明：

- 如果是RDS VPC环境，则必须填写vpc-id和instance-id。
- 如果是普通的MySQL或者经典网络RDS，则vpc-ic和instance-id填写空字符串。
- 目前仅支持部分region的 RDS VPC。支持的region请参考简介。如果您有其他Region的使用需求，请提供单申请。

配置文件参数：

参数	说明
region	您的服务所在区域。
vpc-id	VPC的ID。
instance-id	RDS实例ID。
host	ECS实例ID。
port	ECS实例端口。
username	用户名。
password	密码。
db	数据库。
table	数据表。

配置文件示例：

```
{
    "externalStoreName": "chiji_user",
    "storeType": "rds-vpc",
    "parameter": {
        "vpc-id": "vpc-**********************",
        "instance-id": "rm-**"**************,
```

```
        "host": "rm-*****************.mysql.rds.aliyuncs.com",
        "port": "3306",
        "username": "test****",
        "password": "1234****",
        "db": "chiji",
        "table": "chiji_user",
        "region": "cn-qingdao"
    }
}
```

5. 后续操作。

   您还可以通过CLI来更新或删除外部存储。

   - 更新MySQL外部存储：

     ```
     aliyunlog log update_external_store --project_name="log-rds-demo"
      --config="file:///root/config.json"
     ```

   - 删除MySQL外部存储：

     ```
     aliyunlog log delete_external_store --project_name="log-rds-demo"
      --store_name=abc
     ```

7.7.3 关联OSS数据源

日志服务支持通过外部存储的方式完成OSS和日志服务联合查询，即在查询中将OSS数据与日志服务数据同时作为数据源。

功能优势

- 节约存储成本。对于异构数据，可以根据数据的特性选择合适的存储系统，最大限度的节省成本。 对于更新少的数据，选择存放在OSS上，只需要支付少量的存储费用。如果存放在MySQL上，还要支付计算实例的费用。
- 节约流量费用。OSS是阿里云的存储系统，可以走内网读取数据，免去了流量费用。

前提条件

- 已开通日志服务，并创建了Project和Logstore。

  日志服务准备流程请参考准备流程。
- 已开通OSS，并创建了存储空间。

  详细步骤请参考创建存储空间。

配置步骤

1. 登录OSS控制台，并上传CSV格式文件到OSS。

文件上传步骤请参考上传文件。

2. 在日志服务中定义外部存储。

在日志服务**查询**页面的查询框中输入相应SQL语句，通过SQL定义虚拟外部表，映射到OSS文件。执行结果result为true，表示执行成功。

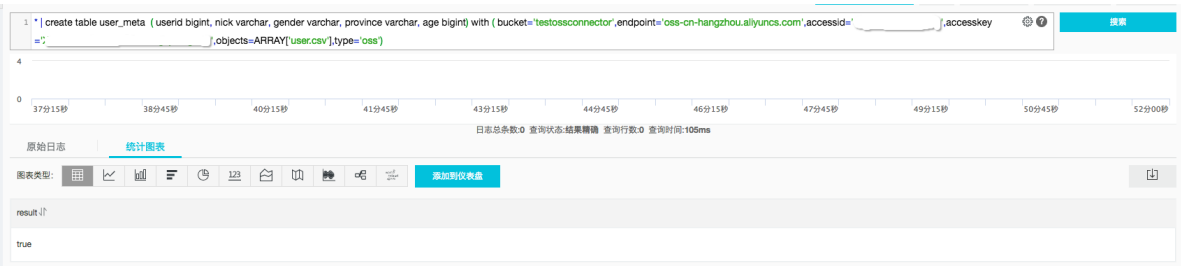
在SQL语句中您需要定义外部存储名称、表的schema等信息，并通过with语法指定OSS访问信息及文件信息。

项目	参数	说明
外部存储名称	*tableName*	外部存储名称，即虚拟表的名称。
表的schema	包括表的列名及格式。	定义表的属性。
OSS访问信息	**endpoint**	地域。
	accessid	您的AccessKeyID。
	accesskey	您的AccessKey Secret。
OSS文件信息	**bucket**	CSV文件所在的OSS Bucket名称。
	objects	CSV文件的路径。 **说明：** **objects**为array类型，可以包含多个OSS文件。
	type	取值固定为oss，表示外部存储类型为OSS。

例如，通过SQL语句定义一个外部存储，名为*tableName*：

```
* | create table tableName ( userid bigint, nick
 varchar, gender varchar, province varchar, gender
 varchar,age bigint) with ( endpoint='oss-cn-hangzhou-
```

```
internal.aliyuncs.com',accessid='**************************',accesskey
 ='****************************',bucket='testoss*********',objects=ARRAY['user
```



在上例中，分别指定了表的属性：

- userid为 bigint类型。
- nick为varchar类型。
- gender为varchar类型。
- province为varchar类型。
- age为bigint类型。

3. 验证是否已成功定义外部存储。

   执行SQL `select * from user_meta`，查看返回结果是否为您之前定义的表内容。

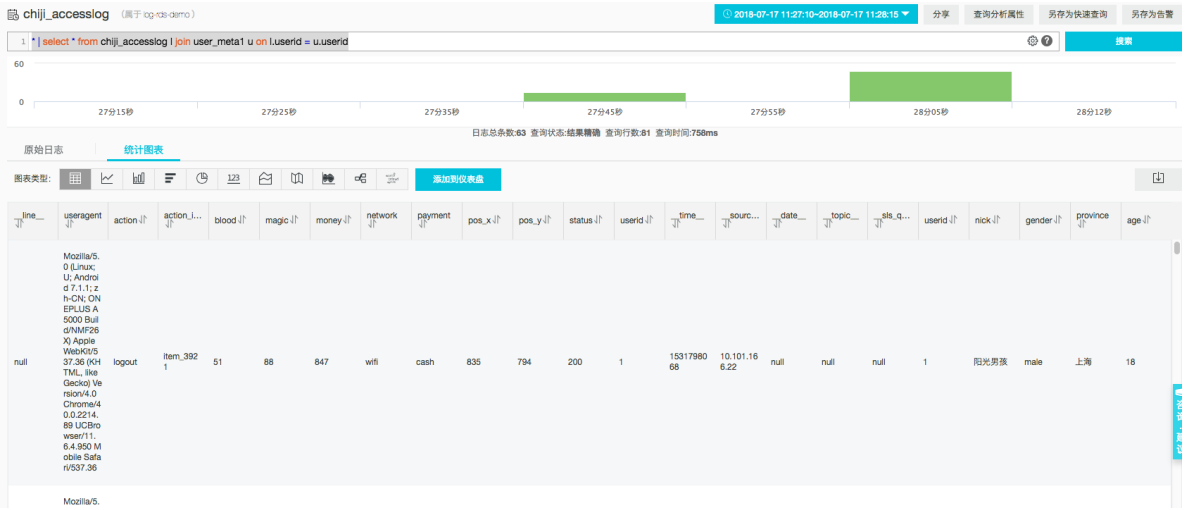


4. 通过JOIN语法完成OSS文件和日志服务数据的联合查询。

   在日志服务**查询**页面，查询框中输入SQL语法，使用JOIN语法，引用OSS外部存储。

   例如，在查询中引用user_meta表，通过关联日志中的id和oss文件中的userid，补全日志的信息。

   ```
   * | select * from chiji_accesslog l join user_meta u on l.userid = u.
   userid
   ```



7.8 分析语法与函数

7.8.1 通用聚合函数

日志服务查询分析功能支持通过通用聚合函数进行日志分析，详细语句及含义如下：

语句	含义	示例
`arbitrary(x)`	随机返回x列中的一个值。	`latency > 100 | select arbitrary(method)`
`avg(x)`	计算x列的算数平均值。	`latency > 100 | select avg(latency)`
`checksum(x)`	计算某一列的checksum，返回base64编码。	`latency > 100 | select checksum(method)`
`count(*)`	计算某一列的个数。	-
`count(x)`	计算某一列非空的个数。	`latency > 100 | count(method)`
`count_if(x)`	计算x=true的个数。	`latency > 100 | count_if(url like '%abc')`
`geometric_mean(x)`	计算某一列的几何平均数。	`latency > 100 | select geometric_mean(latency)`
`max_by(x,y)`	返回当y取最大值时，x当前的值。	查询延时最高的时候，对应的method：`latency>100 | select max_by(method, latency)`

语句	含义	示例
max_by(x,y,n)	返回y最高的n行，对应的x的值。	查询延时最高的3行，对应的method：latency > 100 | select max_by(method,latency,3)
min_by(x,y)	返回当y取最小值时，x当前的值。	查询延时最低的请求，对应的method：* | select min_by(x,y)
min_by(x,y,n)	返回y最小的n行，对应的x的值。	查询延时最小的3行，对应的method： * | select min_by(method,latency,3)
max(x)	返回最大值。	latency > 100| select max(inflow)
min(x)	返回最小值。	latency > 100| select min(inflow)
sum(x)	返回x列的和。	latency > 10 | select sum(inflow)
bitwise_and_agg(x)	对某一列的所有数值做and计算。	-
bitwise_or_agg(x)	对某一列的数值做or计算。	-

7.8.2 安全检测函数

日志服务依托全球白帽子共享安全资产库，提供安全检测函数，您只需要将日志中任意的IP、域名或者URL传给安全检测函数，即可检测是否安全。

应用场景

1. 对服务运维有较强需求的企业和机构如互联网、游戏、资讯等，其IT和安全运维人员可借此以及时筛选可疑访问、攻击以及侵入的行为，并支持进一步深入分析和采取一定措施进行防御。
2. 对内部资产保护有较强需求的企业和机构如银行、证券、电商等，其IT、安全运维人员可以借此即时发现内部访问危险网站、下载木马等行为，并即时采取行动。

功能特点

- 可靠：依托全球共享的白帽子安全资产库，并及时更新。
- 快速：检测百万IP、域名或URL仅需几秒钟。

- 简单：无缝支持任意网络日志，调用3个SQL函数security_check_ip、security_check_domain、security_check_url即可获得结果。
- 灵活：既可以交互式查询，也可以构建报表视图。并可以建立报警并采取进一步行动。

函数列表

函数名	含义	样例
security_check_ip	检查IP是否安全，其中： • 返回1：命中，表示不安全 • 返回0：未命中	`select security_check_ip(real_client_ip)`
security_check_domain	检查Domain是否安全，其中： • 返回1：命中，表示不安全 • 返回0：未命中	`select security_check_domain(site)`
security_check_url	检查URL是否安全，其中： • 返回1：命中，表示不安全 • 返回0：未命中	`select security_check_domain(concat(host, url)`

示例

- **检查外部可疑访问行为并生成报表**

  某电商收集了其运营的Ngnix服务器的日志，对其访问的客户端中想要扫描是否存在不安全的客户IP。可以将Ngnix的日志中的ClientIP字段传给`security_check_ip`函数，并筛选出其返回值为1的IP进行展现，并展示其所在国家、网络运营商等。

对应查询分析语句为：

```
* | select ClientIP, ip_to_country(ClientIP) as country, ip_to_prov
ider(ClientIP) as provider, count(1) as PV where security_check_ip(
ClientIP) = 1 group by ClientIP order by PV desc
```

ClientIP	sec	country	provider	PV
[redacted]	1	中国	电信	575
[redacted]	1	中国	联通	241
[redacted]	1	中国	电信	185
[redacted]	1	中国	联通	179
[redacted]	1	中国	联通	32
[redacted]	1	中国	电信	28

设置为地图视图展示：



- 检查内部可疑访问行为并报警

例如，某证券运营商收集了其内部设备通过网关代理访问外网的网络流量的日志，需要检查是否有人访问了有问题的网站，可以执行如下查询：

```
* | select client_ip, count(1) as PV where security_check_ip(
remote_addr) = 1 or security_check_site(site) = 1 or security_c
heck_url(concat(site, url)) = 1 group by client_ip order by PV desc
```

您也可以将此语句另存为快速查询，并建立安全报警，当有客户端频繁访问危险网站时触发报警，配置每5分钟检查一次是否有人过去1小时内频繁（超过5次）访问危险网站，具体参数可以根据需求调整。配置如下：



7.8.3 Map映射函数

日志服务查询分析功能支持通过映射函数进行日志分析，详细语句及含义如下：

函数	含义	示例
下标运算符[]	获取map中某个key对应的结果。	-
histogram(x)	按照x的每个值GROUP BY，计算count。语法相当于`select count group by x`。  说明： 返回结果为JSON格式。	`latency > 10 | histogram(status)`，等同于`latency > 10 | select count(1) group by status`。
histogram_u(x)	按照x的每个值GROUP BY，计算count。  说明： 返回结果为多行多列。	`latency > 10 | histogram(status)`，等同于`latency > 10 | select count(1) group by status`。
map_agg(Key,Value)	返回Key、Value组成的map，并展示每个method的随机的latency。	`latency > 100 | select map_agg(method,latency)`
multimap_agg(Key,Value)	返回Key、Value组成的多Value map，并返回每个method的所有的latency。	`latency > 100 | select multimap_agg(method, latency)`
cardinality(x) → bigint	获取map的大小。	-
element_at(map<K, V>, key) → V	获取key对应的value。	-
map() → map<unknown, unknown>	返回一个空的map。	-
map(array<K>, array<V>) → map<K,V>	把两个数组，转换成1对1的Map。	`SELECT map(ARRAY[1,3], ARRAY[2,4]); — {1 -> 2, 3 -> 4}`
map_from_entries(array<row<K, V>>) → map<K,V>	把一个多维数组转化成map。	`SELECT map_from_entries(ARRAY[(1, 'x'), (2, 'y')]); — {1 -> 'x', 2 -> 'y'}`
map_entries(map<K, V>) → array<row<K,V>>	把map中的元素转化成array形式。	`SELECT map_entries(MAP(ARRAY[1, 2], ARRAY['x',`

函数	含义	示例
		'y')); — [ROW(1, 'x'), ROW(2, 'y')]
map_concat(map1<K, V>, map2<K, V>, ⋯, mapN<K, V>) → map<K,V>	求多个map的并集，如果某个key在多个map中存在，则取第一个。	-
map_filter(map<K, V>, function) → map<K,V>	请参考lambda map_filter函数。	-
transform_keys(map<K1, V>, function) → MAP<K2,V>	请参考lambda transform_keys函数。	-
transform_values(map<K, V1>, function) → MAP<K,V2>	请参考lambda transform_values函数。	-
map_keys(x<K, V>) → array<K>	获取map中所有的key，返回array。	-
map_values(x<K, V>) → array<V>	获取map中所有的value，返回array。	-
map_zip_with(map<K, V1>, map<K, V2>, function<K, V1, V2, V3>) → map<K,V3>	请参考lambda中 map_zip_with函数。	-

7.8.4 估算函数

日志服务查询分析功能支持通过估算进行日志分析，详细语句及含义如下：

函数	说明	示例
approx_distinct(x)	估算x列的唯一值的个数。	-
approx_percentile(x, percentage)	对于x列排序，找出大约处于percentage位置的数值。	找出位于一半位置的数值：approx_percentile(x,0.5)
approx_percentile(x, percentages)	与上述用法类似，但可以指定多个percentage，找出每个percentage对应的数值。	approx_percentile(x, array[0.1,0.2])
numeric_histogram(buckets, Value)	对于数值列，分多个桶进行统计。即把*Value*一列，分到桶中，桶的个数为*buckets*。	对于POST请求，把延时分为10个桶，查看每个桶的大小：method:POST

函数	说明	示例
	返回内容为每个桶的Key及对应的count数值，相当于针对数值的`select count group by` 说明： 返回结果的格式为JSON。	`| select numeric_histogram(10,latency)`
`numeric_histogram_u(buckets, Value)`	对于数值列，分多个桶进行统计。即把*`Value`*一列，分到桶中，桶的个数为*`buckets`*。 返回内容为每个桶的Key及对应的count数值，相当于针对数值的`select count group by` 说明： 返回结果的格式为多行多列。	对于POST请求，把延时分为10个桶，查看每个桶的大小：`method:POST | select numeric_histogram_u(10,latency)`

7.8.5 数学统计函数

日志服务查询分析功能支持通过数学统计函数进行日志分析，详细语句及含义如下：

语句	含义	示例
`corr(y, x)`	给出两列的相关度，结果从0到1。	`latency>100| select corr(latency,request_size)`
`covar_pop(y, x)`	计算总体协方差。	`latency>100| select covar_pop(request_size,latency)`
`covar_samp(y, x)`	计算样本协方差。	`latency>100| select covar_samp(request_size,latency)`
`regr_intercept(y, x)`	返回输入值的线性回归截距。y是依赖值，x是独立值。	`latency>100| select regr_intercept(request_size,latency)`
`regr_slope(y,x)`	返回输入值的线性回归斜率。y是依赖值，x是独立值。	`latency>100| select regr_slope(request_size,latency)`

语句	含义	示例
stddev(x)或stddev_samp(x)	返回x列的样本标准差。	latency>100| select stddev(latency)
stddev_pop(x)	返回x列的总体标准差。	latency>100| select stddev_pop(latency)
variance(x)或 var_samp(x)	计算x列的样本方差。	latency>100| select variance(latency)
var_pop(x)	计算x列的总体方差。	latency>100| select variance(latency)

7.8.6 数学计算函数

日志服务查询分析功能支持通过数学计算函数进行日志分析，您可以结合查询语句和数学计算函数，对日志查询结果进行数学计算。

数学运算符

数学运算符支持 + - * / % 。可以用在SELECT子句中。

样例：

```
*|select avg(latency)/100 , sum(latency)/count(1)
```

数学计算函数说明

日志服务支持以下运算函数：

函数名	含义
abs(x)	返回x列的绝对值。
cbrt(x)	返回x列的立方根。
ceiling (x)	返回x列向上最接近的整数。
cosine_similarity(x,y)	返回稀疏向量x和y之间的余弦相似度。
degrees	把弧度转化为度。
e()	返回自然常数。
exp(x)	返回自然常数的指数。
floor(x)	返回x向下最接近的整数。
from_base(string,radix)	以radix进制解释string。

函数名	含义
ln(x)	返回自然对数。
log2(x)	返回以2为底，x的对数。
log10(x)	返回以10为底，x的对数。
log(x,b)	返回以b为底，x的对数。
pi()	返回π。
pow(x,b)	返回x的b次幂。
radians(x)	把度转化成弧度。
rand()	返回随机数。
random(0,n)	返回[0，n)随机数。
round(x)	x四舍五入。
round(x, y)	对x保留y个小数为，例如round(1.012345,2) = 1.01。
sqrt(x)	返回x的平方根。
to_base(x, radix)	把x以radix进制表示。
truncate(x)	丢弃掉x的小数部分。
acos(x)	反余弦。
asin(x)	反正弦。
atan(x)	反正切。
atan2(y,x)	y/x的反正切。
cos(x)	余弦。
sin(x)	正弦。
cosh(x)	双曲余弦。
tan(x)	正切。
tanh(x)	双曲正切。
infinity()	double最大值。
is_infinity(x)	判断是否是最大值。
is_finity(x)	判断是否是最大值。
is_nan(x)	判断是否是数值。

7.8.7 字符串函数

日志服务查询分析功能支持通过字符串函数进行日志分析，详细语句及含义如下：

函数名	含义
chr(x)	把int类型转化成对应的unicode字符串，例如chr(65)='A'。
length(x)	字段长度。
levenshtein_distance(string1, string2)	返回两个字符串的最小编辑距离。
lower(string)	转化成小写。
lpad(string, size, padstring)	把string对齐到size大小，如果小于size，用padstring，从左侧补齐到size；如果大于size，则截取到size个。
rpad(string, size, padstring)	类似lpad，从右侧补齐string。
ltrim(string)	删掉左侧的空白字符。
replace(string, search)	把字符串中string中的search删掉。
replace(string, search,rep)	把字符串中string中的search替换为rep。
reverse(string)	翻转string。
rtrim(string)	删掉字符串结尾的空白字符。
split(string,delimeter,limit)	把字符串分裂成array，最多取limit个值。生成的结果为数组，下标从1开始。
split_part(string,delimeter,offset)	把字符串分裂成array，取第offset个字符串。生成的结果为数组，下标从1开始。
split_to_map(string, entryDelimiter, keyValueDelimiter) → map<varchar, varchar>	把string按照entryDelemiter分割成多个entry，每个entry再按照keyValueDelimiter划分成key value。最终返回一个map。
position(substring IN string)	获取string中，substring最先开始的位置。
strpos(string, substring)	查找字符串中的子串的开始位置。返回结果从1开始，如果不存在则返回0。
substr(string, start)	返回字符串的子串，start下标从1开始。
substr(string, start, length)	返回字符串的子串，start下标从1开始。
trim(string)	删掉字符串开头和结尾的空白字符。

函数名	含义
upper(string)	转化为大写字符。
concat(string,string.....)	把两个或多个字符串拼接成一个字符串。
hamming_distance (string1,string2)	获得两个字符串的海明距离。



说明：

字符串需要加单引号包裹，双引号表示列名。例如：a='abc' 表示列a=字符串abc；a='abc' 表示a列=abc列。

7.8.8 日期和时间函数

日志服务支持时间函数、日期函数、区间函数和时序补全函数，您可以在分析语句中使用本文档中介绍的函数。

日期时间类型

1. unixtime：以int类型表示从1970年1月1日开始的秒数，例如1512374067 表示的时间是Mon Dec 4 15:54:27 CST 2017。日志服务每条日志中内置的时间__time__即这种类型。
2. timestamp类型：以字符串形式表示时间，例如2017-11-01 13:30:00。

日期函数

日志服务支持的常见日期函数如下：

函数名	含义	样例
current_date	当天日期。	latency>100| select current_date
current_time	当前时间。	latency>100| select current_time
current_timestamp	结合current_date 和current_time的结果。	latency>100| select current_timestamp
current_timezone()	返回时区。	latency>100| select current_timezone()
from_iso8601_timestamp(string)	把iso8601时间转化成带时区的时间。	latency>100| select from_iso8601_timestamp(iso8601)

函数名	含义	样例
from_iso8601_date(string)	把iso8601转化成天。	latency>100| select from_iso8601_date(iso8601)
from_unixtime(unixtime)	把unix时间转化为时间戳。	latency>100| select from_unixtime(1494985275)
from_unixtime(unixtime, string)	以string为时区，把unixtime转化成时间戳。	latency>100| select from_unixtime (1494985275,'Asia/Shanghai')
localtime	本地时间。	latency>100| select localtime
localtimestamp	本地时间戳。	latency>100| select localtimestamp
now()	等同于current_timestamp。	-
to_unixtime(timestamp)	timestamp转化成unixtime。	*| select to_unixtime('2017-05-17 09:45:00.848 Asia/Shanghai')

时间函数

MySQL时间格式

日志服务还支持MySQL时间格式，包括%a、%b、%y 等。

函数名	含义	样例
date_format(timestamp, format)	把timestamp转化成以format形式表示。	latency>100| select date_format (date_parse ('2017-05-17 09:45:00','%Y-%m-%d %H:%i:%S'), '%Y-%m-%d') group by method
date_parse(string, format)	把string以format格式解析，转化成timestamp。	latency>100|select date_parse('2017-05-17 09:45:00','%Y-%m-%d %H:%i:%S') group by method

表 **7-3:** 格式说明

格式	描述
%a	星期的缩写，即Sun、Sat等。
%b	月份的缩写，即Jan、Dec等。
%c	月份，数值类型，即1~12。
%D	每月的第几天，带后缀，即0th、1st、2nd、3rd等。
%d	每月第几天，十进制格式，范围为01~31。
%e	每月第几天，十进制格式，范围为1~31。
%H	小时，24小时制。
%h	小时，12小时制。
%I	小时，12小时制。
%i	分钟，数值类型，范围为00~59。
%j	每年的第几天，范围为001~366。
%k	小时，范围为0~23。
%l	小时，范围为1~12。
%M	月份的英文表达，范围为January~December。
%m	月份，数值格式，范围为01~12。
%p	AM或PM。
%r	时间，12小时制，格式为`hh:mm:ss AM/PM`。
%S	秒，范围为00~59。
%s	秒，范围为00~59。
%T	时间，24时制，格式为 `hh:mm:ss`。
%U	每年的第几周，星期日是一周的第一天。取值范围为00~53。
%u	每年的第几周，星期一是一周的第一天。范围为00~53。
%V	每年的第几周，星期日是一周的第一天。范围为01~53，与%X同时使用。
%v	每年的第几周，星期一是一周的第一天。范围为01~53，与%x同时使用。
%W	星期几的名称，范围为Sunday到Saturday。

格式	描述
%w	一周的第几天， 星期日为第0天。
%Y	4 位数的年份。
%y	2位数的年份。
%%	%转义字符。

时间段对齐函数

日志服务支持时间段对齐函数，可以按照秒、分钟，小时、日、月、年等对齐。这个函数常用于一些按照时间进行统计的场景。

- 函数语法：

```
date_trunc(unit, x)
```

- 参数说明：

  x 可以是一个timestamp类型，也可以是unix time。

  Unit的取值包括以下类型，其中x取`2001-08-22 03:04:05.000`：

Unit	转化后结果
second	2001-08-22 03:04:05.000
minute	2001-08-22 03:04:00.000
hour	2001-08-22 03:00:00.000
day	2001-08-22 00:00:00.000
week	2001-08-20 00:00:00.000
month	2001-08-01 00:00:00.000
quarter	2001-07-01 00:00:00.000
year	2001-01-01 00:00:00.000

- 示例：

  date_trunc只能在按照一些固定时间间隔统计，如果需要按照灵活的时间维度进行统计（例如统计每5分钟数据），需要按照数学取模方法进行GROUP BY。

```
* | SELECT count(1) as pv,  __time__ - __time__% 300 as minute5gro
upby minute5 limit 100
```

  上述公式中的`%300`表示按照5分钟进行取模对齐。

以下为使用时间格式的一个综合样例。

```
*|select  date_trunc('minute' ,  __time__)  as t,
       truncate (avg(latency) ) ,
       current_date
       group by   t
       order by t  desc
       limit 60
```

时间间隔函数

时间间隔函数用来执行时间段相关的运算，如在日期中添加或减去指定的时间间隔、计算两个日期之间的时间。

函数名	含义	样例
date_add(*unit*, *value*, *timestamp*)	在timestamp上加上value个unit。如果要执行减法，value使用负值。	date_add('day', -7, '2018-08-09 00:00:00') 表示8月9号之前7天
date_diff(*unit*, *timestamp1*, *timestamp2*)	表示timestamp1和timestamp2之间相差几个unit。	date_diff('day', '2018-08-02 00:00:00', '2018-08-09 00:00:00') = 7

该函数支持以下区间单位：

单位	说明
millisecond	毫秒
second	秒
minute	分钟
hour	小时
day	天
week	周
month	月
quarter	季度，即三个月
year	年

时序补全函数

时序补全函数time_series用于处理某些时间缺少的情况。

- 函数格式：

```
time_series(time_column, window, format, padding_data)
```

- 参数说明：

参数	说明
time_column	时间列，例如日志服务提供的默认时间字段__time__。格式为long类型或timestamp类型。
window	窗口大小，由一个数字和单位组成。单位为s（秒）、m（分）、H（小时）、或d（天）。例如2h、5m、3d。
format	MySQL时间格式，表示最终输出的格式。
padding_data	表示补全的内容，包括： • 0：补零。 • null：补null。 • last：补上一个值。 • next：补下一个值。 • avg：补前后的平均值。

- 示例：

  按照每两个小时进行格式化：

```
* | select time_series(__time__, '2h', '%d-%H', '0')  as t, count(1
) group by t order by t
```

  输出：

```
_col1 t
31928 23-04
67546 23-06
```

```
131070 23-08
```

7.8.9 URL函数

URL函数支持从标准URL路径中提取字段，一个标准的URL如下：

```
[protocol:][//host[:port]][path][?query][#fragment]
```

常见URL函数

函数名	含义	样例
`url_extract_fragment(url)`	提取出URL中的fragment，结果为varchar类型。	`*| select url_extract_fragment(url)`
`url_extract_host(url)`	提取出URL中的host，结果为varchar类型。	`*|select url_extract_host(url)`
`url_extract_parameter(url, name)`	提取出URL中的query中name对应的参数值，结果为varchar类型。	`*|select url_extract_parameter(url)`
`url_extract_path(url)`	提取出URL中的path，结果为varchar类型。	`*|select url_extract_path(url)`
`url_extract_port(url)`	提取出URL中的端口，结果为bigint类型。	`*|select url_extract_port(url)`
`url_extract_protocol(url)`	提取出URL中的协议，结果为varchar类型。	`*|select url_extract_protocol(url)`
`url_extract_query(url)`	提取出URL中的query，结果为varchar类型。	`*|select url_extract_query(url)`
`url_encode(value)`	对url进行转义编码。	`*|select url_encode(url)`
`url_decode(value)`	对url进行解码。	`*|select url_decode(url)`

7.8.10 正则式函数

正则式函数解析一串字符串，并且返回需要的一部分子串。

常见的正则式函数及含义如下：

函数名	含义	样例
regexp_extract_all(string, pattern)	返回字符串中命中正则式的所有子串，返回结果是一个字符串数组。	*| SELECT regexp_extract_all('5a 67b 890m', '\d+')，结果为['5','67','890']，*|SELECT regexp_extract_all('5a 67a 890m', '(\d+)a') 结果为['5a','67a']。
regexp_extract_all(string, pattern, group)	返回字符串中命中正则式的第group个()内部分,返回结果是一个字符串数组。	*| SELECT regexp_extract_all('5a 67a 890m', '(\d+)a',1) 结果为['5','67']
regexp_extract(string, pattern)	返回字符串命中的正则式的第一个子串。	*|SELECT regexp_extract('5a 67b 890m', '\d+') 结果为'5'
regexp_extract(string, pattern, group)	返回字符串命中的正则式的弟group个()内的第1个子串。	*|SELECT regexp_extract('5a 67b 890m', '(\d+)([a-z]+)',2) 结果为'b'
regexp_like(string, pattern)	判断字符串是否命中正则式，返回bool类型，正则式可以只命中字符串的一部分。	*|SELECT regexp_like('5a 67b 890m', '\d+m') 结果为true
regexp_replace(string, pattern, replacement)	把字符串中命中正则式的部分替换成replacement。	*|SELECT regexp_replace('5a 67b 890m', '\d+','a') 结果为'aa ab am'
regexp_replace(string, pattern)	把字符串中命中正则式的部分删除，相当于regexp_replace(string,patterm,'')。	*|SELECT regexp_replace('5a 67b 890m', '\d+') 结果为'a b m '
regexp_split(string, pattern)	使用正则式把字符串切分成数组。	*|SELECT regexp_split('5a 67b 890m', '\d+') 结果为['a','b','m']

7.8.11 JSON函数

JSON函数，可以解析一段字符串为JSON类型，并且提取JSON中的字段。JSON主要有两种结构：map和array。如果一个字符串解析成JSON失败，那么返回的是null。

如果需要把json展开成多行，请参考*unnest*语法。

日志服务支持以下常见的JSON函数：

函数名	含义	样例
`json_parse(string)`	把字符串转化成JSON类型。	`SELECT json_parse('[1, 2, 3]')` 结果为JSON类型数组
`json_format(json)`	把JSON类型转化成字符串。	`SELECT json_format( json_parse('[1, 2, 3 ]'))` 结果为字符串
`json_array_contains( json, value)`	判断一个JSON类型数值，或者一个字符串（内容是一个JSON数组）是否包含某个值。	`SELECT json_array _contains(json_parse('[ 1, 2, 3]'), 2)`或 `SELECT json_array_contains('[ 1, 2, 3]', 2)`
`json_array_get( json_array, index)`	同`json_array_contains`，是获取一个JSON数组的某个下标对应的元素。	`SELECT json_array_get ('["a", "b", "c"]', 0 )`结果为`'a'`
`json_array_length(json)`	返回JSON数组的大小。	`SELECT json_array _length('[1, 2, 3]')` 返回结果3
`json_extract(json, json_path)`	从一个JSON对象中提取值，JSON路径的语法类似`$. store.book[0].title`，返回结果是一个JSON对象。	`SELECT json_extract( json, '$.store.book');`
`json_extract_scalar( json, json_path)`	类似`json_extract`，但是返回结果是字符串类型。	-
`json_size(json, json_path)`	获取JSON对象或数组的大小。	`SELECT json_size('[1, 2 , 3]')` 返回结果3

7.8.12 类型转换函数

日志服务配置中支持的类型包括long、double、text类型。在查询中支持的类型包括bigint、double、varchar、timestamp、int等。

类型转换函数，强制把某一列转换成指定类型：

```
cast(value AS type) → type
```

```
try_cast(value AS type) → type
```

7.8.13 IP地理函数

IP 识别函数，可以识别一个IP是内网IP还是外网IP，也可以判断IP所属的国家、省份、城市。

函数名	含义	样例
ip_to_domain(ip)	判断IP所在的域，是内网还是外网。返回intranet或internet。	SELECT ip_to_domain(ip)
ip_to_country(ip)	判断IP所在的国家。	SELECT ip_to_country(ip)
ip_to_province(ip)	判断IP所在的省份。	SELECT ip_to_province(ip)
ip_to_city(ip)	判断IP所在的城市。	SELECT ip_to_city(ip)
ip_to_geo(ip)	判断IP所在的城市的经纬度，范围结果格式为纬度,经度。	SELECT ip_to_geo(ip)
ip_to_city_geo(ip)	判断IP所在的城市的经纬度，返回的是城市经纬度，每个城市只有一个经纬度，范围结果格式为纬度,经度。	SELECT ip_to_city_geo(ip)
ip_to_provider(ip)	获取IP对应的网络运营商。	SELECT ip_to_provider(ip)
ip_to_country(ip,'en')	判断IP所在的国家，返回国际码。	SELECT ip_to_country(ip,'en')
ip_to_country_code(ip)	判断IP所在的国家，返回国际码。	SELECT ip_to_country_code(ip)
ip_to_province(ip,'en')	判断IP所在的省份，返回英文省名或者中文拼音。	SELECT ip_to_province(ip,'en')
ip_to_city(ip,'en')	判断IP所在的城市，返回英文城市名或者中文拼音。	SELECT ip_to_city(ip,'en')

示例

- 在查询中过滤掉内网访问请求，看请求总数

```
* | selectcount(1)whereip_to_domain(ip)!='intranet'
```

- 查看Top10的访问省份

```
* | SELECT count(1) as pv, ip_to_province(ip) as province GROUP BY
province order by pv desc limit 10
```

响应结果样例

```
[
    {
        "__source__": "",
        "__time__": "1512353137",
        "province": "浙江省",
        "pv": "4045"
    }, {
        "__source__": "",
        "__time__": "1512353137",
        "province": "上海市",
        "pv": "3727"
    }, {
        "__source__": "",
        "__time__": "1512353137",
        "province": "北京市",
        "pv": "954"
    }, {
        "__source__": "",
        "__time__": "1512353137",
        "province": "内网IP",
        "pv": "698"
    }, {
        "__source__": "",
        "__time__": "1512353137",
        "province": "广东省",
        "pv": "472"
    }, {
        "__source__": "",
        "__time__": "1512353137",
        "province": "福建省",
        "pv": "71"
    }, {
        "__source__": "",
        "__time__": "1512353137",
        "province": "阿联酋",
        "pv": "52"
    }, {
        "__source__": "",
        "__time__": "1512353137",
        "province": "美国",
        "pv": "43"
    }, {
        "__source__": "",
        "__time__": "1512353137",
```

```
        "province": "德国",
        "pv": "26"
    }, {
        "__source__": "",
        "__time__": "1512353137",
        "province": "吉隆坡",
        "pv": "26"
    }
]
```

在上述结果中包含了内网IP，有时候，开发自己的测试是从内网发出的，为了过滤掉这部分访问请求，可以使用下边的分析语句：

- 过滤掉内网请求，查看Top 10的网络访问省份

```
* | SELECT count(1) as pv, ip_to_province(ip) as province WHERE
ip_to_domain(ip) != 'intranet'  GROUP BY province ORDER BY pv desc
limit 10
```

- 查看不同国家的平均响应延时，最大响应延时，最大延时对应的request

```
* | SELECT AVG(latency),MAX(latency),MAX_BY(requestId, latency) ,
ip_to_country(ip) as country group by country limit 100
```

- 查看不同网络运营商的平均延时

```
* | SELECT AVG(latency) , ip_to_provider(ip) as provider group by
provider limit 100
```

- 查看IP的经纬度，绘制地图

```
* | select count(1) as pv , ip_to_geo(ip) as geo group by geo order
by pv desc
```

返回的格式为：

pv	geo
100	35.3284,-80.7459

7.8.14 GROUP BY 语法

GROUP BY 支持多列。GROUP BY支持通过SELECT的列的别名来表示对应的KEY。

样例：

```
method:PostLogstoreLogs |select avg(latency),projectName,date_trunc('
hour',__time__) as hour   group by projectName,hour
```

别名hour代表第三个SELECT列`date_trunc('hour',__time__)`。这类用法对于一些非常复杂的query非常有帮助。

GROUP BY 支持GROUPING SETS、CUBE、ROLLUP。

样例：

```
method:PostLogstoreLogs |select avg(latency)   group by cube(
projectName,logstore)
method:PostLogstoreLogs |select avg(latency)   group by GROUPING SETS
 ( ( projectName,logstore), (projectName,method))
method:PostLogstoreLogs |select avg(latency)   group by rollup(
projectName,logstore)
```

实践样例

按照时间进行GROUP BY

每条日志都内置了一个时间列`__time__`，当打开任意一列的统计功能后，会自动给时间列打开统计。

使用`date_trunc`函数，可以把时间列对齐到小时(hour)、分钟(minute)、天(day)、月(month)、年(year)。`date_trunc`接受一个对齐单位，和一个unix time或者timestamp类型的列，例如`__time__`。

- 按照每小时、每分钟统计计算PV

```
* | SELECT count(1) as pv , date_trunc('hour',__time__) as hour
group by hour order by hour limit 100
* | SELECT count(1) as pv , date_trunc('minute',__time__) as minute
group by minute order by minute limit 100
```

> 说明：
> limit 100表示最多获取100行，如果不加LIMIT语句，默认最多获取10行数据。

- 按照灵活的时间维度进行统计，例如统计每5分钟的，date_trunc只能在按照一些固定时间间隔统计，这种场景下，我们需要按照数学取模方法进行GROUP BY。

```
* | SELECT count(1) as pv,  __time__ - __time__% 300 as minute5
group by minute5 limit 100
```

上述公式中的`%300`表示按照5分钟进行取模对齐。

在GROUP BY 中提取非agg列

在标准SQL中，如果使用了GROUP BY语法，那么在SELECT时，只能选择SELECT GROUP BY的列原始内容，或者对任意列进行聚合计算，不允许获取非GROUP BY列的内容。

例如，以下语法是非法的，因为b是非GROUP BY的列，在按照a进行GROUP BY时，有多行b可供选择，系统不知道该选择哪一行输出。

```
*|select a, b , count(c) group by a
```

为了达到以上目的，可以使用arbitrary函数输出b：

```
*|select a, arbitrary(b), count(c) group by a
```

7.8.15 窗口函数

窗口函数用来跨行计算。普通的SQL聚合函数只能用来计算一行内的结果，或者把所有行聚合成一行结果。窗口函数，可以跨行计算，并且把结果填到到每一行中。

窗口函数语法：

```
SELECT key1, key2, value,
       rank() OVER (PARTITION BY key2
                    ORDER BY value DESC) AS rnk
FROM orders
ORDER BY key1,rnk
```

核心部分是：

```
rank() OVER (PARTITION BY  KEY1 ORDER BY KEY2 DESC)
```

其中rank()是一个聚合函数，可以使用分析语法中的任何函数，也可以使用本文档列出的函数。PARTITION BY 是值按照哪些桶进行计算。

窗口中使用的特殊聚合函数

函数	含义
rank()	在窗口内，按照某一列排序，返回在窗口内的序号。
row_number()	返回在窗口内的行号。
first_value(x)	返回窗口内的第一个value，一般用法是窗口内数值排序，获取最大值。
last_value(x)	含义和first value相反。
nth_value(x, offset)	窗口内的第offset个数。
lead(x,offset,defaut_value)	窗口内x列某行之后offset行的值，如果不存在该行，则取default_value。

函数	含义
lag(x,offset,defaut_value)	窗口内x列某行之前offset行的值，如果不存在该行，则取default_value。

使用样例

- 在整个公司的人员中，获取每个人的薪水在部门内排名

```
* | select department, persionId, sallary , rank() over(PARTITION
 BY department order by sallary desc) as sallary_rank  order by
department,sallary_rank
```

响应结果：

department	persionId	sallary	sallary_rank
dev	john	9000	1
dev	Smith	8000	2
dev	Snow	7000	3
dev	Achilles	6000	4
Marketing	Blan Stark	9000	1
Marketing	Rob Stark	8000	2
Marketing	Sansa Stark	7000	3

- 在整个公司的人员中，获取每个人的薪水在部门内的占比

```
* | select department, persionId, sallary *1.0 / sum(sallary) over(
PARTITION BY department ) as sallary_percentage
```

响应结果：

department	persionId	sallary	sallary_percentage
dev	john	9000	0.3
dev	Smith	8000	0.26
dev	Snow	7000	0.23
dev	Achilles	6000	0.2
Marketing	Blan Stark	9000	0.375
Marketing	Rob Stark	8000	0.333
Marketing	Sansa Stark	7000	0.29

- 按天统计，获取每天UV相对前一天的增长情况

```
* | select day ,uv, uv *1.0 /(lag(uv,1,0) over() ) as diff_perce
ntage from
(
select approx_distinct(ip) as uv, date_trunc('day',__time__) as day
from log group by day order by day asc
)
```

响应结果：

day	uv	diff_percentage
2017-12-01 00:00:00	100	null
2017-12-02 00:00:00	125	1.25
2017-12-03 00:00:00	150	1.2
2017-12-04 00:00:00	175	1.16
2017-12-05 00:00:00	200	1.14
2017-12-06 00:00:00	225	1.125
2017-12-07 00:00:00	250	1.11

7.8.16 HAVING语法

日志服务查询分析功能支持标准SQL的HAVING语法，和GROUP BY配合使用，用于过滤GROUP BY的结果。

格式：

```
method :PostLogstoreLogs |select avg(latency),projectName group by
projectName  HAVING avg(latency) > 100
```

HAVING和WHERE的区别

HAVING 用于过滤GROUP BY之后的聚合计算的结果，WHERE在聚合计算之间过滤原始数据。

示例

对于气温大于10℃的省份，计算每个省份的平均降雨量，并在最终结果中只显示平均降雨量大于100mL的省份：

```
* | select avg(rain) ,province where temperature > 10 group by
province having avg(rain) > 100
```

7.8.17 ORDER BY语法

ORDER BY 用于对输出结果进行排序，目前只支持按照一列进行排序。

语法格式：

```
orderby  列名 [desc|asc]
```

样例：

```
method :PostLogstoreLogs |select avg(latency) as avg_latency,
projectName group by projectName
HAVING avg(latency) > 5700000
order by avg_latency desc
```

7.8.18 LIMIT语法

LIMIT语法用于限制输出结果的行数。

语法格式

日志服务支持以下两种LIMIT语法格式。

- 只读取前N行：

  ```
  limit N
  ```

- 从S行开始读，读取N行：

  ```
  limit S , N
  ```



说明：

- limit 翻页读取时，只用于获取最终的结果，不可用于获取SQL中间的结果。

- 不支持将limit语法用于子查询内部。例如：

```
* | select count(1) from ( select distinct(url) from limit 0,1000)
```

示例

- 只获取100行结果：

```
* | select distinct(url) from log limit 100
```

- 获取0行到第999行的结果，共计1000行：

```
* | select distinct(url) from log limit 0,1000
```

- 获取第1000行到第1999行的结果，共计1000行：

```
* | select distinct(url) from log limit 1000,1000
```

7.8.19 CASE WHEN和IF分支语法

支持CASE WHEN语法，对连续数据进行归类。例如，从http_user_agent中提取信息，归类成Android和iOS两种类型：

```
SELECT
 CASE
 WHEN http_user_agent like '%android%' then 'android'
 WHEN http_user_agent like '%ios%' then 'ios'
 ELSE 'unknown' END
 as http_user_agent,
    count(1) as pv
    group by  http_user_agent
```

样例

- 计算状态码为200的请求占总体请求的比例：

```
* | SELECT
 sum(
 CASE
 WHEN status =200 then 1
 ELSE 0 end
 ) *1.0 / count(1) as status_200_percentage
```

- 统计不同延时区间的分布：

```
* | SELECT `
 CASE
 WHEN latency < 10 then 's10'
 WHEN latency < 100 then 's100'
 WHEN latency < 1000 then 's1000'
 WHEN latency < 10000 then 's10000'
 else 's_large' end
 as latency_slot,
```

```
  count(1) as pv
  group by latency_slot
```

IF语法

if语法逻辑上等同于CASE WHEN语法。

```
CASE
     WHEN condition THEN true_value
     [ ELSE false_value ]
 END
```

- if(condition, true_value)

  如果condition是true，则返回true_value这一列，否则返回null。

- if(condition, true_value, false_value)

  如果condition是true，则返回true_value这一列，否则返回false_value这一列。

COALESCE语法

coalesce 返回多个列的第一个非Null值。

```
coalesce(value1, value2[,...])
```

NULLIF 语法

如果value1和value2相等，返回null，否则返回value1。

```
nullif(value1, value2)
```

TRY 语法

try语法可以捕获一些底层的异常，例如除0错误，返回null值。

```
try(expression)
```

7.8.20 嵌套子查询

针对一些复杂的查询场景，一层SQL无法满足需求，通过SQL嵌套查询可以满足复杂的需求。

嵌套子查询和无嵌套查询的区别在于，要在SQL中指定from 条件。在查询中要指定`from log`这个关键字，表示从日志中读取原始数据。

样例：

```
* | select sum(pv) from
(
select count(1) as pv from log group by method
```

```
)
```

7.8.21 数组

语句	含义	示例
下标运算符[]	[]用于获取数组中的某个元素。	-

连接运算符||	||用于把两个数组连接成一个数组。	`SELECT ARRAY [1] || ARRAY [2]; — [1, 2]`
		`SELECT ARRAY [1] || 2; — [1, 2]`
		`SELECT 2 || ARRAY [1]; — [2, 1]`

array_distinct	数组去重，获取数组中的唯一元素。	-
array_intersect(x, y)	获取x，y两个数组的交集。	-
array_union(x, y) → array	获取x，y两个数组的并集。	-
array_except(x, y) → array	获取x，y两个数组的差集	-
array_join(x, delimiter, null_replacement) → varchar	把字符串数组用delimiter连接，拼接成字符串，null值用null_replacement替代。 说明： 使用array_join函数时，返回结果最大为1 KB，超出1 KB的数据会被截断。	-
array_max(x) → x	获取x中的最大值。	-
array_min(x) → x	获取x中的最小值。	-
array_position(x, element) → bigint	获取element在x中的下标，下标从1开始。如果找不到，则返回0。	-
array_remove(x, element) → array	从数组中移除element。	-
array_sort(x) → array	给数组排序，null值放到最后。	-
cardinality(x) → bigint	获取数组的大小。	-

语句	含义	示例
concat(array1, array2, ⋯, arrayN) → array	连接数组。	-
contains(x, element) → boolean	如果x中包含element，则返回true。	-
filter(array, function) → array	function 是一个Lambda函数，请参考*lambda*函数中的filter。	-
flatten(x) → array	把二维的array拼接成一维的array。	-
reduce(array, initialState, inputFunction, outputFunction) → x	请参考*lambda*函数reduce。	-
reverse(x) → array	把x反向排列。	-
sequence(start, stop) → array	生成从start到stop结束的一个序列，每一步加1。	-
sequence(start, stop, step) → array	生成从start到stop结束的一个序列，每一步加step。	-
sequence(start, stop, step) → array	start和stop是timestamp类型，生成从start到stop结束的timestamp数组。step是INTERVAL类型，可以是DAY到SECOND，也可以是YEAR或MONTH。	-
shuffle(x) → array	重新随机分布array。	-
slice(x, start, length) → array	获取x数组从start开始，length个元素组成新的数组。	-
transform(array, function) → array	请参考*lambda*函数transform()。	-
zip(array1, array2[, ⋯]) → array	合并多个数组。结果的第M个元素的第N个参数，是原始第N个数组的第M个元素，相当于把多个数组进行了转置。	`SELECT zip(ARRAY[1, 2], ARRAY['1b', null, '3b']);` — `[ROW(1, '1b'), ROW(2, null), ROW(null, '3b')]`

语句	含义	示例
zip_with(array1, array2, function) → array	请参考*lambda*函数zip_with。	-
array_agg (key)	array_agg (key)是一个聚合函数，表示把key这一列的所有内容变成一个array返回。	`* | select array_agg(key)`

7.8.22 二进制字符串函数

二进制字符串类型varbinary有别于字符串类型varchar。

语句	说明
连接函数 ||	`a || b` 结果为ab。
length(binary) → bigint	返回二进制的长度。
concat(binary1, …, binaryN) → varbinary	连接二进制字符串，等同于||。
to_base64(binary) → varchar	把二进制字符串转换成base64。
from_base64(string) → varbinary	把base64转换成二进制字符串。
to_base64url(binary) → varchar	转化成url安全的base64。
from_base64url(string) → varbinary	从url安全的base64转化成二进制字符串。
to_hex(binary) → varchar	把二进制字符串转化成十六进制表示。
from_hex(string) → varbinary	从十六进制转化成二进制。
to_big_endian_64(bigint) → varbinary	把数字转化成大端表示的二进制。
from_big_endian_64(binary) → bigint	把大端表示的二进制字符串转化成数字。
md5(binary) → varbinary	计算二进制字符串的md5。
sha1(binary) → varbinary	计算二进制字符串的sha1。
sha256(binary) → varbinary	计算二进制字符串的sha256 hash。
sha512(binary) → varbinary	计算二进制字符串的sha512。
xxhash64(binary) → varbinary	计算二进制字符串的xxhash64。

7.8.23 位运算

语句	说明	示例

bit_count(x, bits) → bigint	统计x的二进制表示中，1的个数。	`SELECT bit_count(9, 64); — 2`
		`SELECT bit_count(9, 8); — 2`
		`SELECT bit_count(-7, 64); — 62`
		`SELECT bit_count(-7, 8); — 6`

bitwise_and(x, y) → bigint	以二进制的形式求x,y的and的值。	-
bitwise_not(x) → bigint	以二进制的形式求对x的所有位取反。	-
bitwise_or(x, y) → bigint	以二进制形式对x，y求or。	-
bitwise_xor(x, y) → bigint	以二进制形式对x，y求xor。	-

7.8.24 同比和环比函数

同比和环比函数用于比较当前区间的计算结果和之前一个指定区间的结果。

函数	含义	样例
`compare(value, time_window)`	表示将当前时段计算出来的value值和time_window计算出来的结果进行比较。 value为double或long类型，time_window单位为秒；返回值为数组类型。 返回值分别是当前值、time_window之前的值和当前值与之前值的比值。	`*|select compare( pv , 86400) from (select count(1) as pv from log )`
`compare(value, time_window1, time_window2)`	表示当前区间分别和time_window1和time_window2之前的区间值进行比较，结果为json数组。其中，各个值的大小必须满足以下规则：[当前值, time_window1之前的值，	`* | select compare(pv, 86400, 172800) from ( select count(1) as pv from log)`

函数	含义	样例
	time_window2之前的值，当前值/time_window1之前的值, 当前值/time_window2之前的值]。	
compare(value, time_window1, time_window2, time_window3)	表示当前区间分别和time_window1和time_window2,time_window3之前的区间值进行比较，结果为json数组。其中，各个值的大小必须满足以下规则：[当前值, time_window1之前的值，time_window2之前的值，time_window3之前的值，当前值/time_window1之前的值, 当前值/time_window2之前的值,当前值/time_window3之前的值]。	* | select compare(pv, 86400, 172800,604800) from (select count(1) as pv from log)
compare_result(value, time_window)	效果等同于compare(value,time_window)，但返回结果是字符串类型，格式为"当前值(增长百分比%)"。其中增长百分比默认保留2位小数。	* | select compare_result(pv, 86400) from (select count(1) as pv from log)
compare_result(value,time_window1, time_window2)	效果等同于compare(value,time_window1, time_window2)，但返回结果是字符串类型，格式为"当前值(相比第一个区间增长百分比%) (相比第二个区间增长百分比)"。其中增长百分比默认保留2位有效小数。	* | select compare_result(pv, 86400,172800) from (select count(1) as pv from log)
ts_compare(value, time_window)	表示当前区间分别和time_window1和time_window2之前的区间值进行比较，结果为json数组。其中，各个值的大小必须遵循以下规则：[当前值, time_window1之前的值，当前	例如，* | select t, ts_compare(pv, 86400) as d from(select date_trunc('minute', __time__) as t, count(1) as pv from log group by t order by t) group by t表示将当前时间段每分钟

函数	含义	样例
	值/time_window1之前的值，前一个时间七点的unix时间戳]。用于时序函数比较，需要在SQL中对时间列进行GROUP BY。	的计算结果和上一个时间段每分钟的计算结果进行比较。结果为：`d:[1251.0,1264.0, 0.9897151898734177, 1539843780.0,1539757380.0]t:2018-10-19 14:23:00.000`。

示例

- 计算当前1小时和昨天同一时段的PV比例。

  开始时间为2018-7-25 14:00:00；结束时间为2018-07-25 15:00:00。

  查询分析语句：

  ```
  * | select compare( pv , 86400) from (select count(1) as pv from log
  )
  ```

  其中，86400表示当前时段减去86400秒。

  返回结果：

  ```
  [9.0,19.0,0.47368421052631579]
  ```

  其中，

  - 9.0表示从2018-7-25 14:00:00到2018-07-25 15:00:00的PV值。
  - 19.0表示2018-7-24 14:00:00到2018-07-24 15:00:00的PV值。
  - 0.47368421052631579表示当前时段与之前时段的比值。

  如果要把数组展开成3列数字，分析语句为：

  ```
  * | select diff[1],diff[2],diff[3] from(select compare( pv , 86400)
  as diff from (select count(1) as pv from log))
  ```

- 计算当前1小时内每分钟的PV和昨天同时段的PV比值，并以折线图展示。

  1. 计算当前1小时内每分钟的PV和昨天同时段的PV比值。开始时间为2018-7-25 14:00:00，结束时间为2018-07-25 15:00:00。

查询分析语句：

```
*| select t, compare( pv , 86400) as diff from (select count(1) as
 pv, date_format(from_unixtime(__time__), '%H:%i') as t from log
group by t) group by t order by t
```

返回结果：

t	diff
14:00	[9520.0,7606.0,1.2516434393899554]
14:01	[8596.0,8553.0,1.0050274757395066]
14:02	[8722.0,8435.0,1.0340248962655603]
14:03	[7499.0,5912.0,1.2684370771312586]

其中t表示时间，格式为小时:分钟。diff列的内容是一个数组，分别表示：

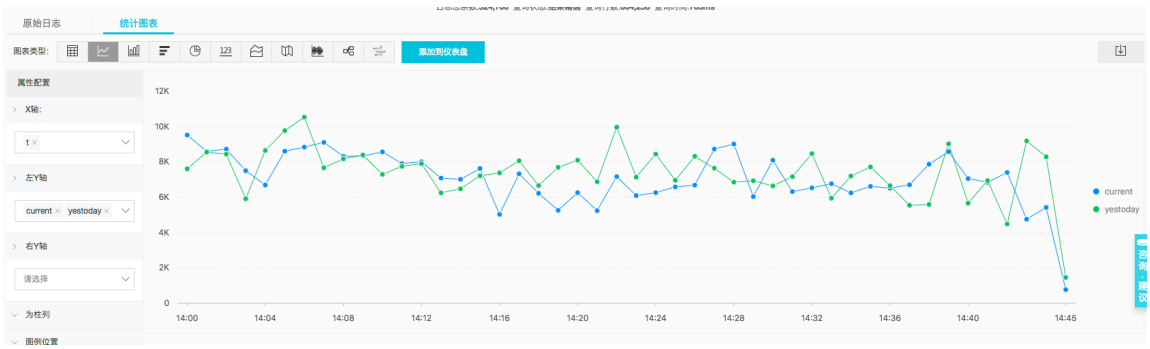
- 当前时段的PV值。
- 之前时段的PV值。
- 当前时段PV值与之前时段比值。

2. 通过以下语句将查询结果展开为折线图形式：

```
*|select t, diff[1] as current, diff[2] as yestoday, diff[3] as
 percentage from(select t, compare( pv , 86400) as diff from (
select count(1) as pv, date_format(from_unixtime(__time__), '%H:%i
') as t from log group by t) group by t order by t)
```

将查询结果配置为折线图，两条线分别表示今天的值和昨天的值：

图 **7-15:** 折线图



7.8.25 比较函数和运算符

比较函数和运算符

比较运算判断参数的大小关系，可以应用于任何可比较类型，如int、bigint、double和text等。

比较运算符

比较运算符用于比较两个参数值的大小关系。当用比较运算符比较两个值时，如果逻辑成立，则返回true；否则返回false。

运算符	含义
<	小于
>	大于
<=	小于或等于
>=	大于或等于
=	等于
<>	不等于
!=	不等于

范围运算符 **BETWEEN**

BETWEEN用于判断一个参数的值是否在另外两个参数之间，范围为闭区间。

- 如果逻辑成立，则返回true；否则返回false。

  示例：`SELECT 3 BETWEEN 2 AND 6;`逻辑成立，返回true。

  以上样例等同于`SELECT 3 >= 2 AND 3 <= 6;`。
- BETWEEN可以跟在not之后，用于相反逻辑的判断。

  示例：`SELECT 3 NOT BETWEEN 2 AND 6;`，逻辑不成立，返回false。

  以上样例等同于`SELECT 3 < 2 OR 3 > 6;`。
- 如果三个参数中任何一个包含Null，则返回的结果为Null。

IS NULL 和 IS NOT NULL

该运算符用于判断参数是否是Null值。

IS DISTINCT FROM 和 IS NOT DISTINCT FROM

类似于相等和不等判断，区别在于该运算符能够判断存在NULL值的情况。

样例：

```
SELECT NULL IS DISTINCT FROM NULL; -- false
SELECT NULL IS NOT DISTINCT FROM NULL; -- true
```

如下表所示，DISTINCT运算符可以判断多种情况下的参数大小关系。

a	b	a = b	a <> b	a DISTINCT b	a NOT DISTINCT b
1	1	TRUE	FALSE	FALSE	TRUE
1	2	FALSE	TRUE	TRUE	FALSE
1	NULL	NULL	NULL	TRUE	FALSE
NULL	NULL	NULL	NULL	FALSE	TRUE

GREATEST 和 LEAST

用于获取多列中的最大值或者最小值。

示例：

```
select greatest(1,2,3) ; -- 返回3
```

比较判断：ALL、ANY 和 SOME

比较判断用于判断参数是否满足条件。

- ALL用于判断参数是否满足所有条件。如果逻辑成立，则返回true，否则返回false。
- ANY用于判断参数是否满足条件之一。如果逻辑成立，则返回true，否则返回false。
- SOME和ANY一样，用于判断参数是否满足条件之一。
- ALL、ANY 和 SOME必须紧跟在比较运算符之后。

如下表所示，ALL和ANY支持多种情况下的比较判断。

表达式	含义
A = ALL (…)	A等于所有的值时，结果才是true。
A <> ALL (…)	A不等于所有的值时，结果才是true。
A < ALL (…)	A小于所有的值时，结果才是true。

表达式	含义
A = ANY (…)	A等于任何一个值时，结果就为true，等同于 A IN (…)。
A <> ANY (…)	A不等于任何一个值时，结果为true。
A < ANY (…)	A小于其中最大值时，结果为true。

示例：

```
SELECT 'hello' = ANY (VALUES 'hello', 'world'); -- true
SELECT 21 < ALL (VALUES 19, 20, 21); -- false
SELECT 42 >= SOME (SELECT 41 UNION ALL SELECT 42 UNION ALL SELECT 43);
 -- true
```

7.8.26 lambda函数

Lambda表达式

lambda表达式的书写形式为`->`。

样例：

```
x -> x + 1
(x, y) -> x + y
x -> regexp_like(x, 'a+')
x -> x[1] / x[2]
x -> IF(x > 0, x, -x)
x -> COALESCE(x, 0)
x -> CAST(x AS JSON)
x -> x + TRY(1 / 0)
```

大多数的MySQL表达式都可以在lambda中使用。

filter(array<T>, function<T, boolean>) → ARRAY<T>

从一个array中过滤数据，只取满足function返回true的元素。

示例：

```
SELECT filter(ARRAY [], x -> true); -- []
SELECT filter(ARRAY [5, -6, NULL, 7], x -> x > 0); -- [5, 7]
SELECT filter(ARRAY [5, NULL, 7, NULL], x -> x IS NOT NULL); -- [5, 7]
```

map_filter(map<K, V>, function<K, V, boolean>) → MAP<K,V>

从map中过滤数据，只取满足function返回true的元素对。

示例：

```
SELECT map_filter(MAP(ARRAY[], ARRAY[]), (k, v) -> true); -- {}
SELECT map_filter(MAP(ARRAY[10, 20, 30], ARRAY['a', NULL, 'c']), (k, v
) -> v IS NOT NULL); -- {10 -> a, 30 -> c}
SELECT map_filter(MAP(ARRAY['k1', 'k2', 'k3'], ARRAY[20, 3, 15]), (k,
v) -> v > 10); -- {k1 -> 20, k3 -> 15}
```

reduce(array<T>, initialState S, inputFunction<S, T, S>, outputFunction<S, R>) → R

reduce函数，从初始状态开始，依次遍历array中的每一个元素，每次在状态S的基础上，计算inputFunction(s,t)，生成新的状态。最终应用outputFunction，把最终状态S变成输出结果R。

1. 初始状态S
2. 遍历每个元素T。
3. 计算inputFunction(S,T)，生成新状态S。
4. 重复2、3，直到最后一个元素被遍历以及生成新状态。
5. 利用最终状态S，获取最终输出结果R。

示例：

```
SELECT reduce(ARRAY [], 0, (s, x) -> s + x, s -> s); -- 0
SELECT reduce(ARRAY [5, 20, 50], 0, (s, x) -> s + x, s -> s); -- 75
SELECT reduce(ARRAY [5, 20, NULL, 50], 0, (s, x) -> s + x, s -> s);
 -- NULL
SELECT reduce(ARRAY [5, 20, NULL, 50], 0, (s, x) -> s + COALESCE(x, 0
), s -> s); -- 75
SELECT reduce(ARRAY [5, 20, NULL, 50], 0, (s, x) -> IF(x IS NULL, s, s
 + x), s -> s); -- 75
SELECT reduce(ARRAY [2147483647, 1], CAST (0 AS BIGINT), (s, x) -> s
 + x, s -> s); -- 2147483648
SELECT reduce(ARRAY [5, 6, 10, 20], -- calculates arithmetic average:
10.25
              CAST(ROW(0.0, 0) AS ROW(sum DOUBLE, count INTEGER)),
              (s, x) -> CAST(ROW(x + s.sum, s.count + 1) AS ROW(sum
DOUBLE, count INTEGER)),
              s -> IF(s.count = 0, NULL, s.sum / s.count));
```

transform(array<T>, function<T, U>) → ARRAY<U>

对数组中的每个元素，依次调用function，生成新的结果U。

示例：

```
SELECT transform(ARRAY [], x -> x + 1); -- []
SELECT transform(ARRAY [5, 6], x -> x + 1); -- [6, 7] 表示对每个元素执行
加1操作
SELECT transform(ARRAY [5, NULL, 6], x -> COALESCE(x, 0) + 1); -- [6,
1, 7]
SELECT transform(ARRAY ['x', 'abc', 'z'], x -> x || '0'); -- ['x0', '
abc0', 'z0']
```

```
SELECT transform(ARRAY [ARRAY [1, NULL, 2], ARRAY[3, NULL]], a ->
filter(a, x -> x IS NOT NULL)); -- [[1, 2], [3]]
```

transform_keys(map<K1, V>, function<K1, V, K2>) → MAP<K2,V>

依次对map中的每个key应用函数，生成新的key。

示例：

```
SELECT transform_keys(MAP(ARRAY[], ARRAY[]), (k, v) -> k + 1); -- {}
SELECT transform_keys(MAP(ARRAY [1, 2, 3], ARRAY ['a', 'b', 'c']), (k
, v) -> k + 1); -- {2 -> a, 3 -> b, 4 -> c}  表示对每个key执行加1操作。
SELECT transform_keys(MAP(ARRAY ['a', 'b', 'c'], ARRAY [1, 2, 3]), (k
, v) -> v * v); -- {1 -> 1, 4 -> 2, 9 -> 3}
SELECT transform_keys(MAP(ARRAY ['a', 'b'], ARRAY [1, 2]), (k, v) -> k
 || CAST(v as VARCHAR)); -- {a1 -> 1, b2 -> 2}
SELECT transform_keys(MAP(ARRAY [1, 2], ARRAY [1.0, 1.4]), -- {one ->
1.0, two -> 1.4}
                      (k, v) -> MAP(ARRAY[1, 2], ARRAY['one', 'two'])[
k]);
```

transform_values(map<K, V1>, function<K, V1, V2>) → MAP<K, V2>

对map中的所有value应用function函数，把V1变成V2，生成新的map<K,V2>。

```
SELECT transform_values(MAP(ARRAY[], ARRAY[]), (k, v) -> v + 1); -- {}
SELECT transform_values(MAP(ARRAY [1, 2, 3], ARRAY [10, 20, 30]), (k,
v) -> v + 1); -- {1 -> 11, 2 -> 22, 3 -> 33}
SELECT transform_values(MAP(ARRAY [1, 2, 3], ARRAY ['a', 'b', 'c']), (
k, v) -> k * k); -- {1 -> 1, 2 -> 4, 3 -> 9}
SELECT transform_values(MAP(ARRAY ['a', 'b'], ARRAY [1, 2]), (k, v) -
> k || CAST(v as VARCHAR)); -- {a -> a1, b -> b2}
SELECT transform_values(MAP(ARRAY [1, 2], ARRAY [1.0, 1.4]), -- {1 ->
one_1.0, 2 -> two_1.4}
                        (k, v) -> MAP(ARRAY[1, 2], ARRAY['one', 'two
'])[k] || '_' || CAST(v AS VARCHAR));
```

zip_with(array<T>, array<U>, function<T, U, R>) → array<R>

合并两个array，通过函数指定生成的新的array中的元素。第一个数组的元素T和第二个数组元素U，生成新的结果R。

示例：

```
SELECT zip_with(ARRAY[1, 3, 5], ARRAY['a', 'b', 'c'], (x, y) -> (y,
x)); --表示调换前后两个数组的元素位置，生成一个新的数组。结果： [ROW('a', 1),
ROW('b', 3), ROW('c', 5)]
SELECT zip_with(ARRAY[1, 2], ARRAY[3, 4], (x, y) -> x + y); -- 结果[4,
6]
```

```
SELECT zip_with(ARRAY['a', 'b', 'c'], ARRAY['d', 'e', 'f'], (x, y) ->
concat(x, y)); 表示把前后两个数组的元素拼接，生成一个新的字符串。结果： ['ad',
 'be', 'cf']
```

map_zip_with(map<K, V1>, map<K, V2>, function<K, V1, V2, V3>) → map<K, V3>

合并两个map，针对每个key，由两个value V1和V2生成V3。生成新的Map<K,V3>。

```
SELECT map_zip_with(MAP(ARRAY[1, 2, 3], ARRAY['a', 'b', 'c']),
                    MAP(ARRAY[1, 2, 3], ARRAY['d', 'e', 'f']),
                    (k, v1, v2) -> concat(v1, v2));  表示把两个map key相
同的value进行合并。-- {1 -> ad, 2 -> be, 3 -> cf}
SELECT map_zip_with(MAP(ARRAY['k1', 'k2'], ARRAY[1, 2]),
                    MAP(ARRAY['k2', 'k3'], ARRAY[4, 9]),
                    (k, v1, v2) -> (v1, v2));表示把两个value生成一个数
组。-- {k1 -> ROW(1, null), k2 -> ROW(2, 4), k3 -> ROW(null, 9)}
SELECT map_zip_with(MAP(ARRAY['a', 'b', 'c'], ARRAY[1, 8, 27]),
                    MAP(ARRAY['a', 'b', 'c'], ARRAY[1, 2, 3]),
                    (k, v1, v2) -> k || CAST(v1/v2 AS VARCHAR)); 表示在
结果中连接key的值和两个value的相除结果-- {a -> a1, b -> b4, c -> c9}
```

7.8.27 逻辑函数

逻辑运算符

表 **7-4:** 逻辑运算符

运算符	描述	样例
AND	只有左右运算数都是true时，结果才为true	a AND b
OR	左右运算数任一个为true，结果为true	a OR b
NOT	右侧运算数为false时，结果才为true	NOT a

NULL参与逻辑运算

a和b分别取值TRUE FALSE和NULL时的真值表如下：

表 **7-5:** 真值表**1**

a	**b**	**a AND b**	**a OR b**
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE

a	b	a AND b	a OR b
TRUE	NULL	NULL	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE
FALSE	NULL	FALSE	NULL
NULL	TRUE	NULL	TRUE
NULL	FALSE	FALSE	NULL
NULL	NULL	NULL	NULL

表 7-6: 真值表2

a	NOT a
TRUE	FALSE
FALSE	TRUE
NULL	NULL

7.8.28 列的别名

在SQL标准中，列名必须由字母、数字、下划线组成，且以字母开头。

如果在日志收集配置中，用户如果配置了不符合SQL标准的列名(例如User-Agent)，那么需要在配置统计属性的页面，给列取一个别名，用于查询。别名仅仅用于SQL统计，在底层存储时，仍然是原始名称，搜索时需要使用原始名称。

此外，当用户原始的列名特别长时，也可以取一个别名来代替原始列名查询。

表 7-7: 别名样例

原始列名	别名
User-Agent	ua
User.Agent	ua
123	col
abceefghijklmnopqrstuvw	a

7.8.29 Logstore和RDS联合查询

背景信息

日志服务支持Logstore中的日志数据和RDS数据库进行联合查询，以及把查询结果保存到RDS中。

操作步骤

1. 创建RDS VPC，并设置白名单。

   a) 创建RDS，并指定VPC环境。创建成功后，得到VPC ID和RDS实例ID。

   b) 设置RDS白名单：`100.104.0.0/16`。

   详细步骤请参考《*RDS*用户指南》中**设置白名单**章节。

2. 创建External Store。

   通过以下语句创建External Store，请将参数替换为您的实际参数值。

```
{
"externalStoreName":"storeName",
"storeType":"rds-vpc",
"parameter":
   {
   "region":"cn-qingdao",
   "vpc-id":"vpc-m5eq4irc1pucp*******"
   "instance-id":"i-m5eeo2whsn*******"
   "host":"localhost",
   "port":"3306",
   "username":"root",
   "password":"****",
   "db":"scmc"
   "table":"join_meta"
   }
}
```

表 7-8: 参数说明

参数	说明
region	您的服务所在区域。
vpc-id	VPC的ID。
instance-id	RDS实例ID。
host	ECS实例ID。
port	ECS实例端口。
username	用户名。
password	密码。

参数	说明
db	数据库。
table	数据表。



说明：

目前仅支持北京（cn-beijing）、青岛（cn-qingdao）和杭州（cn-hangzhou）区域。

3. Join查询。

   在日志服务控制台**查询**页面执行Join语句。

   支持的Join语法：

   - INNER JOIN
   - LEFT JOIN
   - RIGHT JOIN
   - FULL JOIN

```
[ INNER ] JOIN
LEFT [ OUTER ] JOIN
RIGHT [ OUTER ] JOIN
FULL [ OUTER ] JOIN
```



说明：

- Join仅支持Logstore Join小表。
- 在Join顺序中，Logstore必须写在前部，External Store写在后部。
- Join中必须写External Store的名称，会自动替换成RDS的db+表名。不能直接填写RDS表名。

   Join语法样例：

```
method:postlogstorelogs | select count(1) , histogram(logstore) from
 log  l join join_meta m on l.projectid = cast( m.ikey as varchar)
```

4. 保存查询结果到RDS中。

支持通过Insert语法把查询结果插入到RDS中。

```
method:postlogstorelogs | insert into method_output  select cast(
methodasvarchar(65535)),count(1)fromloggroupbymethod
```

Python 程序样例

```
# encoding: utf-8
from __future__ import print_function
from aliyun.log import *
from aliyun.log.util import base64_encodestring
from random import randint
import time
import os
from datetime import datetime
    endpoint = os.environ.get('ALIYUN_LOG_SAMPLE_ENDPOINT',
 'cn-chengdu.log.aliyuncs.com')
    accessKeyId = os.environ.get('ALIYUN_LOG_SAMPLE_ACCESSID
', '')
    accessKey = os.environ.get('ALIYUN_LOG_SAMPLE_ACCESSKEY
', '')
    logstore = os.environ.get('ALIYUN_LOG_SAMPLE_LOGSTORE',
 '')
    project = "ali-yunlei-chengdu"
    client = LogClient(endpoint, accessKeyId, accessKey,
token)
#创建external store
    res = client.create_external_store(project,ExternalSt
oreConfig("rds_store","region","rds-vpc","vpc id","实例id
","实例ip","实例端口","用户名","密码","数据库","数据库表"));
    res.log_print()
    #获取external store详情
    res = client.get_external_store(project,"rds_store");
    res.log_print()
    res = client.list_external_store(project,"");
    res.log_print();
    # JOIN查询
    req = GetLogStoreLogsRequest(project,logstore,From,To
,"","select count(1) from  "+ logstore +"  s join  meta m on
  s.projectid = cast(m.ikey as varchar)");
    res = client.get_logs(req)
    res.log_print();
     # 查询结果写入RDS
    req = GetLogStoreLogsRequest(project,logstore,From,
To,"","," insert into rds_store select count(1) from  "+
logstore );
    res = client.get_logs(req)
    res.log_print();
```

7.8.30 空间几何函数

空间几何概念

空间几何函数支持Well-Known Text (WKT) 格式描述的几何实体。

表 **7-9:** 几何实体格式

几何实体	**Well-Known Text (WKT)** 格式
点	`POINT (0 0)`
线段	`LINESTRING (0 0, 1 1, 1 2)`
多边形	`POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1))`
多点	`MULTIPOINT (0 0, 1 2)`
多线段	`MULTILINESTRING ((0 0, 1 1, 1 2), (2 3, 3 2, 5 4))`
多个多边形	`MULTIPOLYGON (((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1)), ((-1 -1, -1 -2, -2 -2, -2 -1, -1 -1)))`
空间实体集合	`GEOMETRYCOLLECTION (POINT(2 3), LINESTRING (2 3, 3 4))`

构造空间实体

表 **7-10:** 构造空间实体函数说明

函数	说明
ST_Point(double, double) → Point	构造一个点。
ST_LineFromText(varchar) → LineString	从WKT格式的文本中构造一个线段。
ST_Polygon(varchar) → Polygon	从WKT格式的文本中构造一个多边形。
ST_GeometryFromText(varchar) → Geometry	从WKT文本中构造一个空间几何实体。
ST_AsText(Geometry) → varchar	把一个空间几何实体转变成WKT格式。

运算符

函数	说明
ST_Boundary(Geometry) → Geometry	计算几何实体的闭包。
ST_Buffer(Geometry, distance) → Geometry	返回一个多边形，该多边形距离输入参数Geometry的距离是distance。

函数	说明
ST_Difference(Geometry, Geometry) → Geometry	返回两个空间实体的不同的点的集合。
ST_Envelope(Geometry) → Geometry	返回空间实体的边界多边形。
ST_ExteriorRing(Geometry) → Geometry	返回多边形的外部环。
ST_Intersection(Geometry, Geometry) → Geometry	返回两个空间实体的交集点。
ST_SymDifference(Geometry, Geometry) → Geometry	返回两个空间实体不同的点，组成的新的空间实体。获取两个几何对象不相交的部分。

空间关系判断

函数	说明
ST_Contains(Geometry, Geometry) → boolean	当第二个实体的所有点都不在第一个实体外部，并且第一个实体至少有一个内部点在第二个实体内部时，返回true。如果第二个实体正好在第一个实体的边上，那么是false。
ST_Crosses(Geometry, Geometry) → boolean	当两个实体有共同内部点时，返回true。
ST_Disjoint(Geometry, Geometry) → boolean	当两个实体没有任何交集时，返回true。
ST_Equals(Geometry, Geometry) → boolean	当两个实体完全相同时，返回true。
ST_Intersects(Geometry, Geometry) → boolean	当两个实体在两个空间上共享时，返回true。
ST_Overlaps(Geometry, Geometry) → boolean	当两个实体维度相同，并且不是包含关系时，返回true。
ST_Relate(Geometry, Geometry) → boolean	当两个实体相关时，返回true。
ST_Touches(Geometry, Geometry) → boolean	当两个实体仅仅边界有联系，没有共同内部点时，返回true。
ST_Within(Geometry, Geometry) → boolean	当第一个实体完全在第二个实体内部时，返回true。如果边界有交集，返回false。

Accessors

函数	说明
ST_Area(Geometry) → double	使用欧几里得测量法，计算多边形在二维平面上的投影面积。
ST_Centroid(Geometry) → Geometry	返回几何实体的中心点。
ST_CoordDim(Geometry) → bigint	返回几何实体的坐标维度。
ST_Dimension(Geometry) → bigint	返回几何实体的固有维度，必须小于或等于坐标维度。
ST_Distance(Geometry, Geometry) → double	计算两个实体之间的最小距离。
ST_IsClosed(Geometry) → boolean	当实体时一个闭合空间时，返回true。
ST_IsEmpty(Geometry) → boolean	当参数时一个空的几何实体集合或者多边形或者点时返回true。
ST_IsRing(Geometry) → boolean	当参数是一条线，并且时闭合的简单的线时，返回true。
ST_Length(Geometry) → double	在二维投影平面上，使用欧几里得测量法计算一个线段或者多条线段的长度。返回一个行字符串或多行字符串的长度。该长度是采用欧几里得测量法基于空间参考对二维平面的预测。
ST_XMax(Geometry) → double	返回几何体边框的X最大值。
ST_YMax(Geometry) → double	返回几何体边框的Y最大值。
T_XMin(Geometry) → double	返回几何体边框的X最小值。
ST_YMin(Geometry) → double	返回结合体边框的Y最小值。
ST_StartPoint(Geometry) → point	返回线段类型几何体的第一个点。
ST_EndPoint(Geometry) → point	返回线段类型几何体的最后一个点。
ST_X(Point) → double	返回点类型的X轴。
ST_Y(Point) → double	返回点类型的Y轴。
ST_NumPoints(Geometry) → bigint	计算几何实体的点的个数。
ST_NumInteriorRing(Geometry) → bigint	返回多边形内部的环的个数。

7.8.31 地理函数

IP转国家、省、城市、运营商、经纬度，请参考文档*IP*地理函数。

表 7-11: 地理函数

函数名	含义	样例
geohash(string)	将纬度、经度用geohash编码，string为字符串类型，内容是维度、逗号、经度。	`select geohash('34.1,120.6')= 'wwjcbrdnzs'`
geohash(lat, lon)	将纬度、经度用geohash编码，参数分别是纬度和经度。	`select geohash(34.1,120.6)= 'wwjcbrdnzs'`

7.8.32 Join语法

Join用于多表中字段之间的联系。日志服务除了支持单个Logstore的Join之外，还支持Logstore和RDS的Join，以及Logstore和Logstore的Join。本文档为您介绍如何使用跨Logstore的Join功能。

操作步骤

1. 下载最新版本Python SDk。
2. 使用GetProjectLogs接口进行查询。

SDK示例

```
#!/usr/bin/env python
#encoding: utf-8
import time,sys,os
from aliyun.log.logexception import LogException
from aliyun.log.logitem import LogItem
from aliyun.log.logclient import LogClient
from aliyun.log.getlogsrequest import GetLogsRequest
from aliyun.log.getlogsrequest import GetProjectLogsRequest
from aliyun.log.putlogsrequest import PutLogsRequest
from aliyun.log.listtopicsrequest import ListTopicsRequest
from aliyun.log.listlogstoresrequest import ListLogstoresRequest
from aliyun.log.gethistogramsrequest import GetHistogramsRequest
from aliyun.log.index_config import *
from aliyun.log.logtail_config_detail import *
from aliyun.log.machine_group_detail import *
from aliyun.log.acl_config import *
if __name__=='__main__':
    token = None
    endpoint = "http://cn-hangzhou.log.aliyuncs.com"
    accessKeyId  = 'LTAIvKy7U'
    accessKey='6gXLNTLyCfdsfwrewrfhdskfdsfuiwu'
    client = LogClient(endpoint, accessKeyId, accessKey,token)
    logstore = "meta"
    # 在查询语句中，指定两个Logstore，每个Logstore要分别指定各自的时间范围，以及两个Logstore关联的key
    req = GetProjectLogsRequest(project,"select count(1) from sls_operation_log  s join  meta m on s.__date__ >'2018-04-10 00:00:00' and s.__date__ < '2018-04-11 00:00:00' and m.__date__ >'2018-04-23 00:00:00' and m.__date__ <'2018-04-24 00:00:00' and s.projectid = cast(m.ikey as varchar)");
```

```
    res = client.get_project_logs(req)
    res.log_print();
    exit(0)
```

7.8.33 unnest语法

应用场景

处理数据时，一列数据通常为字符串或数字等primitive类型的数据。在复杂的业务场景下，日志数据的某一列可能会是较为复杂的格式，例如数组（array）、对象(map)、JSON等格式。对这种特殊格式的日志字段进行查询分析，可以使用unnest语法。

例如以下日志：

```
__source__:  1.1.1.1
__tag__:__hostname__:  vm-req-170103232316569850-tianchi111932.tc
__topic__:  TestTopic_4
array_column:  [1,2,3]
double_column:  1.23
map_column:  {"a":1,"b":2}
text_column:  商品
```

其中`array_column`字段为数组类型。如果统计`array_column`中所有数值的汇总值，需要遍历每一行的数组中的每一个元素。

unnest语法结构

语法	说明
`unnest( array) as table_alias( column_name)`	表示把array类型展开成多行，行的名称为`column_name`。
`unnest(map) as table(key_name, value_name)`	表示把map类型展开成多行，key的名称为`key_name`，value的名称为`value_name`。



说明：

注意，由于unnest接收的是array或者map类型的数据，如果您的输入为字符串类型，那么要先转化成json类型，然后再转化成array类型或map类型，转化的方式为`cast(json_parse( array_column) as array(bigint))`。

遍历数组每一个元素

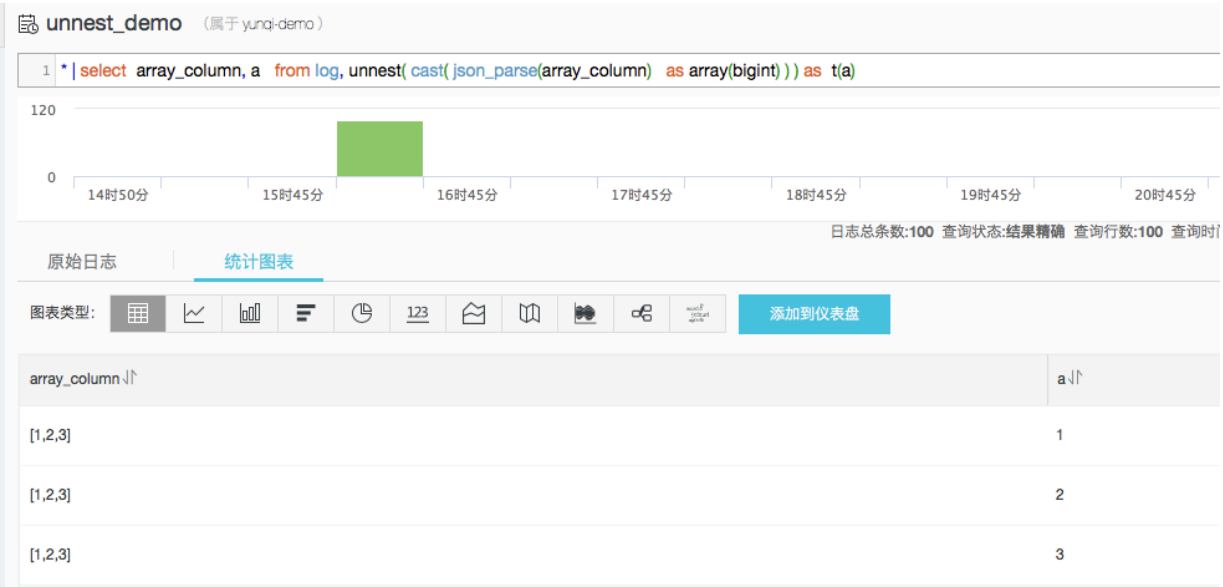
使用SQL把array展开成多行：

```
* | select  array_column, a   from log, unnest( cast( json_parse(
array_column)   as array(bigint) ) ) as  t(a)
```

上述SQL把数组展开成多行数字，`unnest( cast( json_parse(array_column) as array(bigint) ) ) as t(a)`，unnest语法把数组展开，以t来命名新生成的表，使用a来引用展开后的列。

结果如下图：

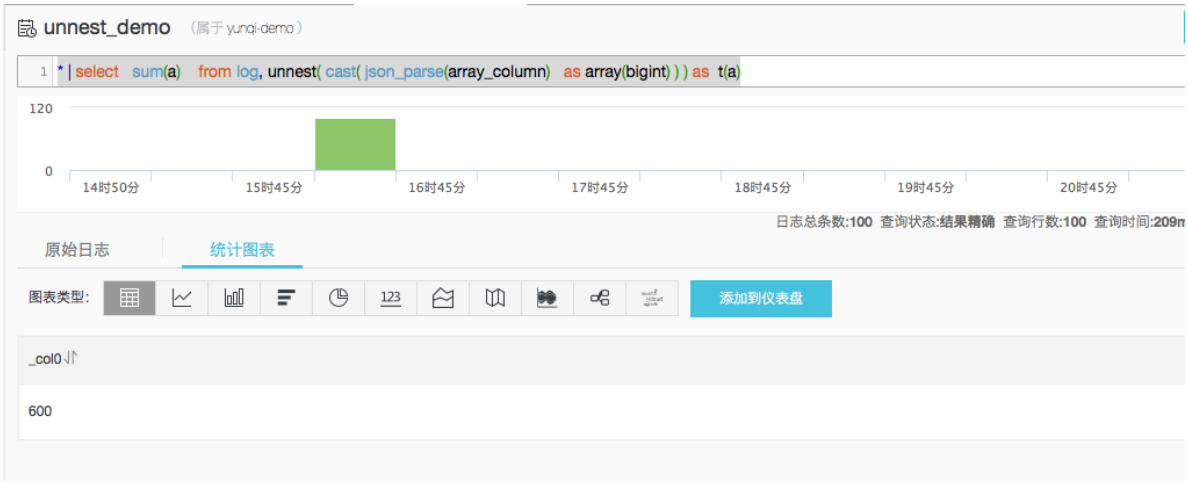
图 **7-16:** 展开数组



- 统计数组中的每个元素的和：

```
* | select   sum(a)    from log, unnest( cast( json_parse(array_colu
mn)   as array(bigint) ) ) as  t(a)
```

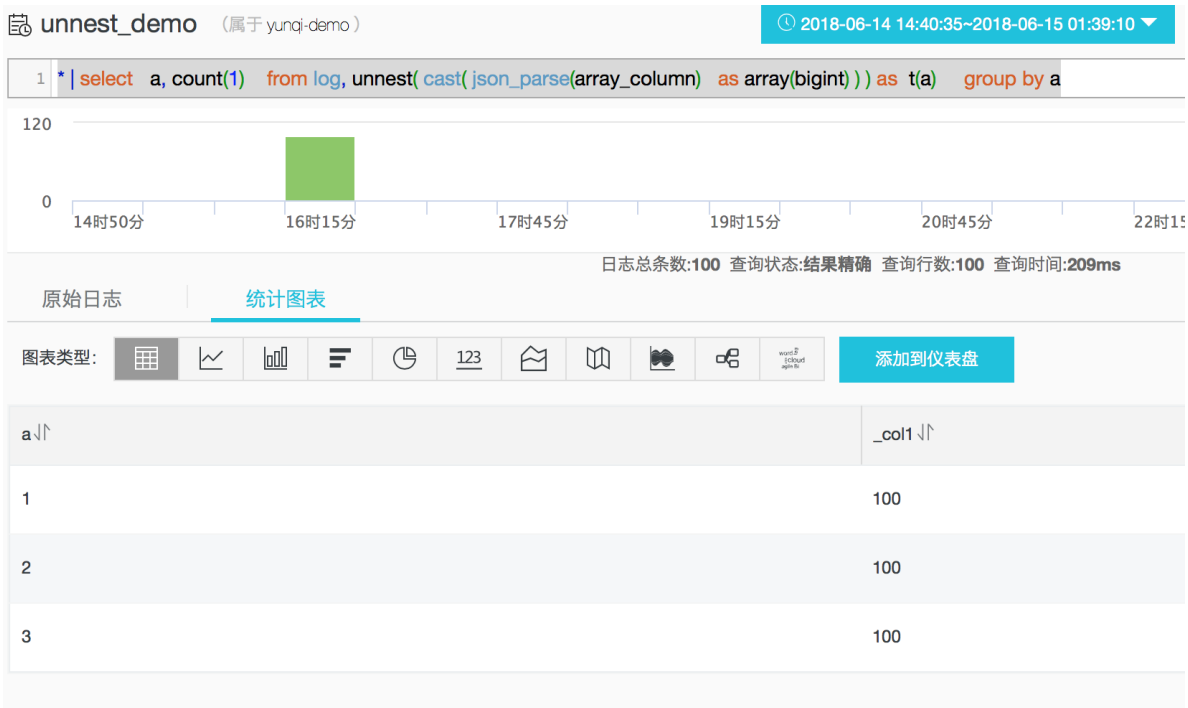
图 7-17: 对数组进行sum计算



- 按照数组中的每个元素进行group by计算：

```
* | select   a, count(1)    from log, unnest( cast( json_parse(
array_column)   as array(bigint) ) ) as  t(a)     group by a
```

图 7-18: 对数组进行group by计算

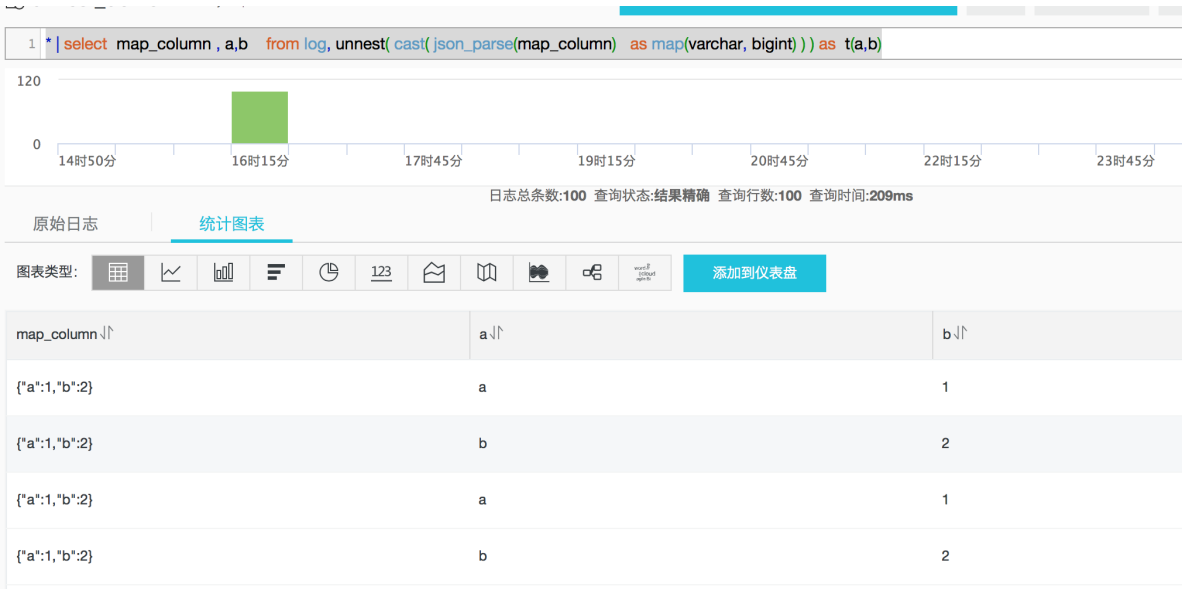


遍历Map

- 遍历Map中的元素：

```
* | select  map_column , a,b    from log, unnest( cast( json_parse(
map_column)   as map(varchar, bigint) ) ) as  t(a,b)
```

图 7-19: 遍历Map



- 按照Map的key进行group by 统计：

```
* | select   key,  sum(value)    from log, unnest( cast( json_parse
(map_column)   as map(varchar, bigint) ) ) as  t(key,value)    GROUP
  BY  key
```

图 7-20: 对Key进行group by统计



格式化显示**histogram,numeric_histogram**的结果

- histogram

  histogram函数类似于count group by 语法。语法请参考*Map*映射函数。

  通常情况下histogram的结果为一串json数据，无法配置视图展示，例如：

```
* | select histogram(method)
```

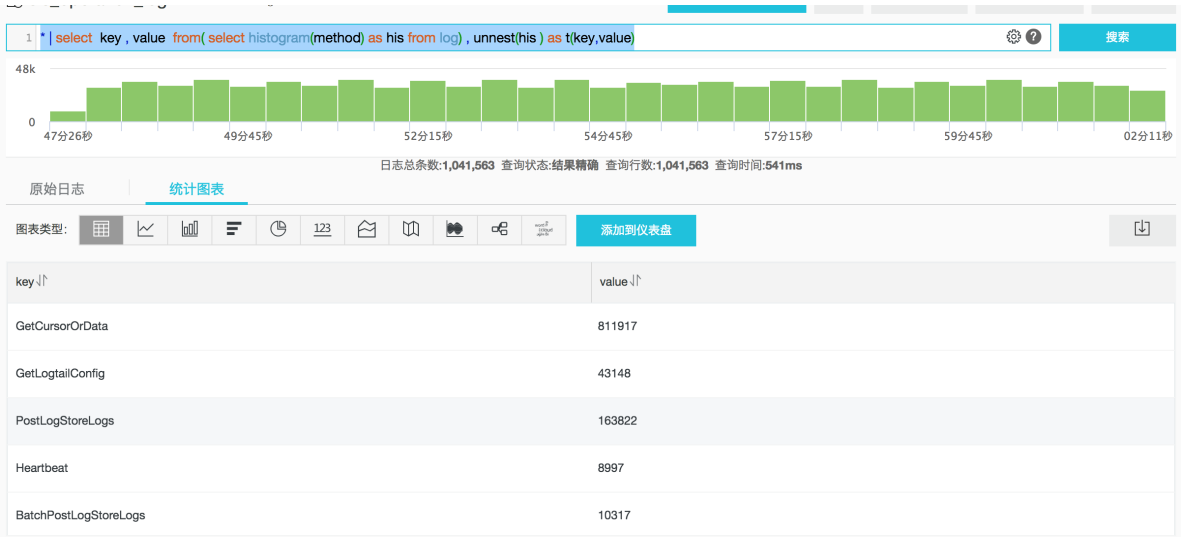
图 **7-21:** 普通**histogram**结果



您可以通过unnest语法，把JSON展开成多行配置视图，例如：

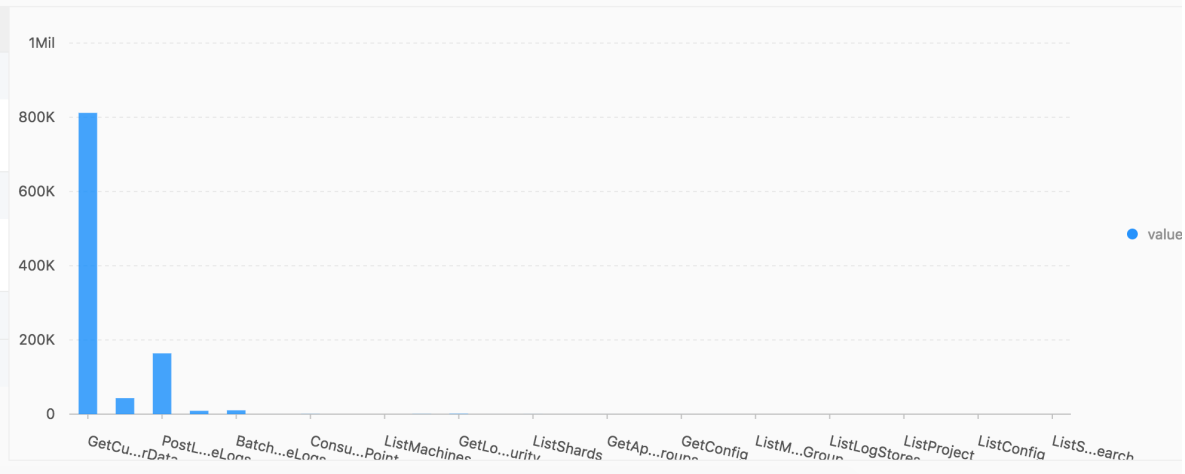
```
* | select  key , value  from( select histogram(method) as his from log) , unnest(his ) as t(key,value)
```

图 **7-22:** 展开**JSON**



接下来，可以配置可视化视图：

图 **7-23:** 可视化视图**1**



- numeric_histogram

  numeric_histogram语法是为了把数值列分配到多个桶中去，相当于对数值列进行group by，具体语法请参考估算函数。

```
* | select numeric_histogram(10,Latency)
```

numeric_histogram的输出如下：

图 **7-24: numeric_histogram**查询结果

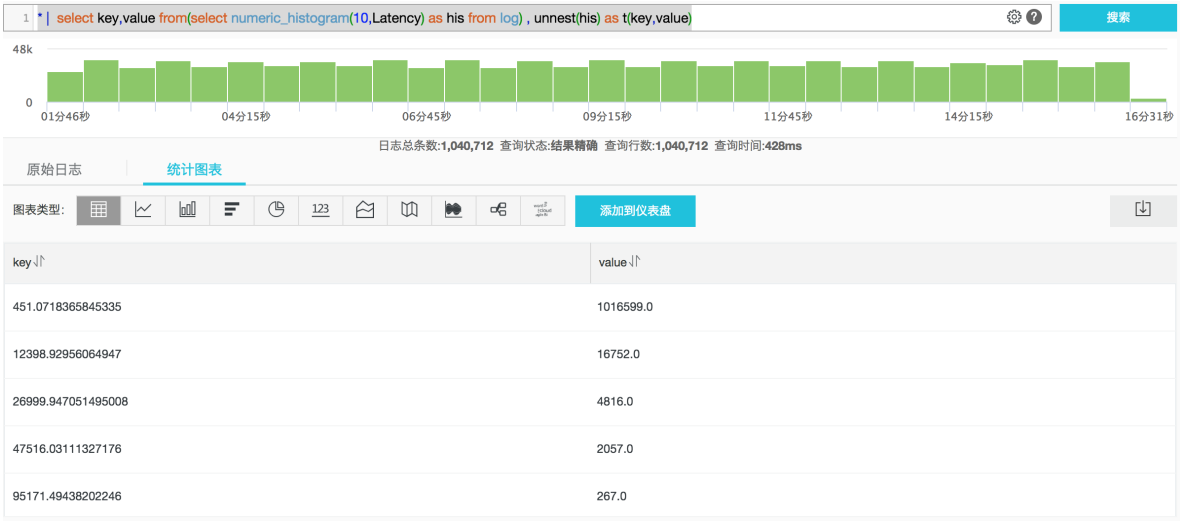


您可以通过以下查询语句格式化展示该结果：

```
* |  select key,value from(select numeric_histogram(10,Latency) as
his from log) , unnest(his) as t(key,value)
```

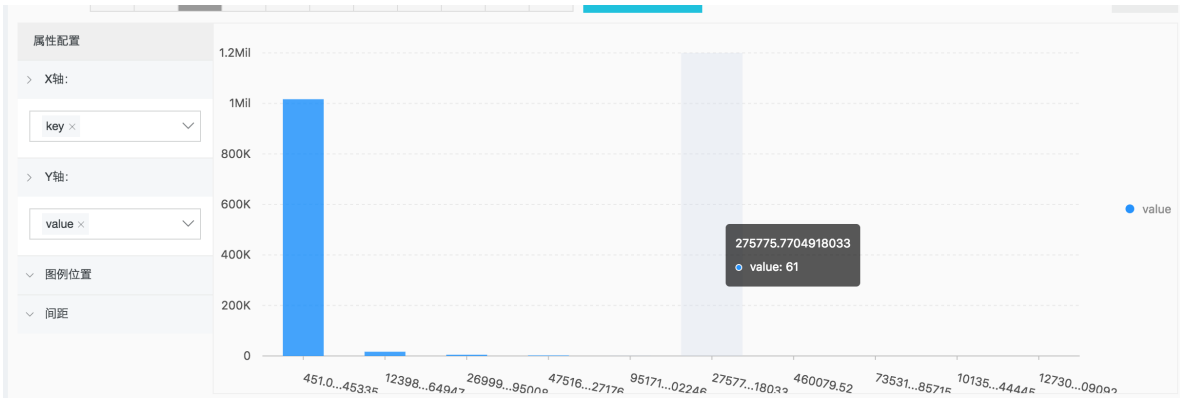
结果如下：

图 7-25: 查询结果



同时配置柱状图的形式展示：

图 7-26: 可视化视图2



7.9 机器学习语法与函数

7.9.1 简介

日志服务（Log Service）机器学习功能为您提供多种功能丰富的算法和便捷的调用方式，您可以在日志查询分析中通过SELECT语句和机器学习函数调用机器学习算法，分析某一字段或若干字段在一段时间内的特征。

尤其是针对时序数据分析场景，日志服务提供了丰富的时序分析算法，可以帮助您快速解决时序预测、时序异常检测、序列分解、多时序聚类等场景问题，兼容SQL标准接口，大大降低了您使用算法的门槛，提高分析问题和解决问题的效率。

功能特点

- 支持单时序序列的多种平滑操作。
- 支持单时序序列的预测、异常检测、变点检测、折点检测、多周期估计算法。
- 支持单时序序列的分解操作。
- 支持多时序序列的多种聚类算法。
- 支持多字段（数值列、文本列）的模式挖掘。

限制说明

- 输入的时序数据必须是基于相同时间间隔的采样数据。
- 输入的时序数据中不能含有重复时间点的数据。

限制项	说明
时序数据处理的有效容量	上限为150,000个连续时间点数据。 若数量超过上限，请进行聚合操作或者降采样。
密度聚类算法的聚类容量	上限为5000条时序曲线，每条时序曲线的长度最大为1440个点。
层次聚类算法的聚类容量	上限为2000条时序曲线，每条时序曲线的长度最大为1440个点。

机器学习函数

	类别	函数	说明
时间序列	平滑函数	ts_smooth_simple	使用Holt Winters算法对时序数据平滑。
		ts_smooth_fir	使用FIR滤波器对时序数据平滑。
		ts_smooth_iir	使用IIR滤波器对时序数据平滑。
	多周期估计函数	ts_period_detect	对时序数据进行分段周期估计。
	变点检测函数	ts_cp_detect	寻找时序序列中具有不同统计特性的区间，区间端点即为变点。

	类别	函数	说明
		ts_breakout_detect	寻找时序序列中，某统计量发生陡升或陡降的点。
	预测与异常检测函数	ts_predicate_simple	利用默认参数对时序数据进行建模，并进行简单的时序预测和异常点的检测。
		ts_predicate_ar	使用自回归模型对时序数据进行建模，并进行简单的时序预测和异常点的检测。
		ts_predicate_arma	使用移动自回归模型对时序数据进行建模，并进行简单的时序预测和异常点检测。
		ts_predicate_arima	使用带有差分的移动自回归模型对时序数据进行建模，并进行简单的时序预测和异常点检测。
	序列分解函数	ts_decompose	使用STL算法对时序数据进行序列分解。
	时序聚类函数	ts_dencity_cluster	使用密度聚类方法对多条时序数据进行聚类。
		ts_hierarchical_cluster	使用层次聚类方法对多条时序数据进行聚类。
		ts_similar_instance	查找到指定曲线名称的相似曲线。
模式挖掘	频繁模式统计	pattern_stat	统计模式中的频繁模式，在给定的多属性字段样本中，挖掘出具有一定代表性的属性组合。
	差异模式统计	pattern_diff	在指定条件下找出导致两个集合差异的模式。

7.9.2 平滑函数

平滑函数是针对输入的时序曲线进行平滑和简单的滤波操作，滤波操作通常是发现时序曲线形态的第一步。

函数列表

函数	说明
ts_smooth_simple	默认平滑函数，使用Holt Winters算法对时序数据平滑。
ts_smooth_fir	使用FIR滤波器对时序数据平滑。
ts_smooth_iir	使用IIR滤波器对时序数据平滑

ts_smooth_simple

函数格式：

```
select ts_smooth_simple(x, y)
```

参数说明如下：

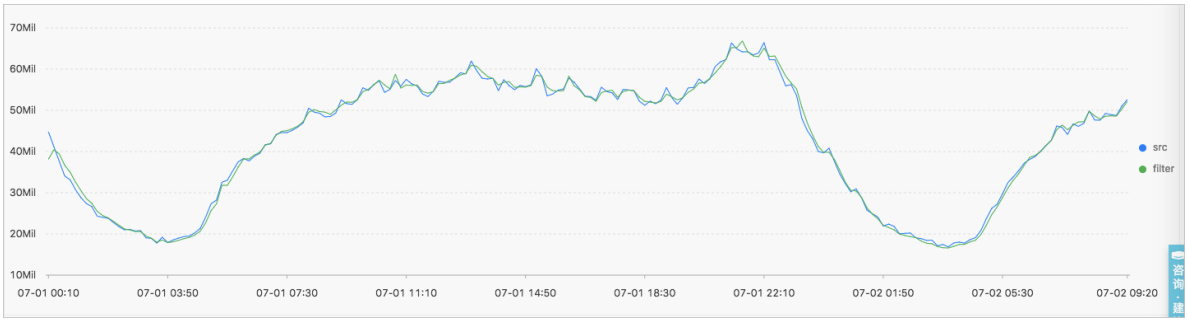
参数	说明	取值
x	时间列，顺序为从小到大。	Unixtime时间戳，单位为秒。
y	数值列，对应某时刻的数据。	--

示例：

- 查询分析：

```
* | select ts_smooth_simple(stamp, value) from ( select __time__ - __time__ % 120 as stamp, avg(v) as value from log GROUP BY stamp order by stamp )
```

- 输出结果：



显示项如下：

显示项		说明
横轴	unixtime	数据的时间戳，单位为秒。
纵轴	src	未滤波前的数据。
	filter	滤波之后的数据。

ts_smooth_fir

函数格式：

- 当您无法确定滤波参数时，可以使用内置窗口的参数进行滤波操作：

```
select ts_smooth_fir(x, y,winType,winSize,samplePeriod,sampleMethod)
```

- 若您可以确定滤波参数，可以根据需求自定义设置滤波参数：

```
select ts_smooth_fir(x, y,array[],samplePeriod,sampleMethod)
```

参数说明如下：

参数	说明	取值
x	时间列，从小到大排列。	格式为Unixtime时间戳，单位为秒。
y	数值列，对应某时刻的数据。	-
winType	滤波的窗口类型。	取值包括： • rectangle：矩形窗口。 • hanning：汉宁窗。 • hamming：汉明窗。 • blackman：布莱克曼窗。 说明： 推荐您选择rectangle类型以获得更好的显示效果。
winSize	滤波窗口的长度。	long类型，取值范围为2~15。
array[]	FIR滤波的具体参数。	格式为数组，例如array[0.2, 0.4, 0.3, 0.1]。
samplePeriod	对当前时序数据进行采样的周期。	long类型，单位为秒。取值范围为1~86399。

参数	说明	取值
sampleMethod	针对采样窗口内数据的采样方法。	取值包括： • avg：表示取窗口内数据的平均值。 • max：表示取窗口内数据的最大值。 • min：表示取窗口内数据的最小值。 • sum：表示取窗口内数据的总和。

示例：

- 查询分析：

```
* | select ts_smooth_fir(stamp, value, 'rectangle', 4, 1, 'avg')
from ( select __time__ - __time__ % 120 as stamp, avg(v) as value
from log GROUP BY stamp order by stamp )
```

输出结果：



- 查询分析：

```
* | select ts_smooth_fir(stamp, value, array[0.2, 0.4, 0.3, 0.1], 1,
 'avg') from ( select __time__ - __time__ % 120 as stamp, avg(v) as
value from log GROUP BY stamp order by stamp )
```

输出结果：



显示项如下：

显示项		说明
横轴	unixtime	数据的Unixtime时间戳，单位为秒。
纵轴	src	未滤波前的数据。
	filter	滤波之后的数据。

ts_smooth_iir

函数格式：

```
select
 ts_smooth_iir(x, y, array[], array[], samplePeriod, sampleMethod)
```

参数说明如下：

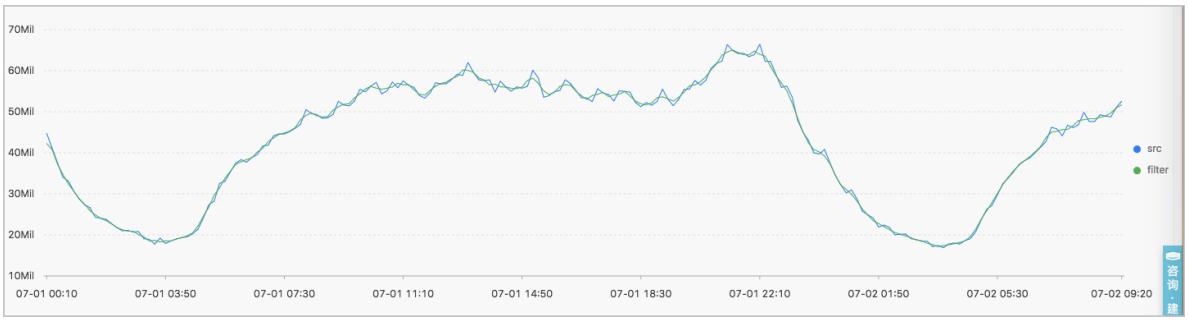
参数	说明	取值
x	时间列，从小到大排列。	格式为Unixtime时间戳，单位为秒。
y	数值列，对应某时刻的数据。	-
array[]	IIR滤波算法中关于x_i的具体参数。	数组格式，长度（length）的取值范围为2~15。例如array[0.2, 0.4, 0.3, 0.1]。
array[]	IIR滤波算法中关于y_{i-1}的具体参数。	数组格式，长度（length）的取值范围为2~15。例如array[0.2, 0.4, 0.3, 0.1]。
samplePeriod	对当前时序数据进行采样的周期。	long类型，单位为秒。取值范围为1~86399。
sampleMethod	针对采样窗口内数据的采样方法。	取值包括： • avg：表示取窗口内数据的平均值。 • max：表示取窗口内数据的最大值。 • min：表示取窗口内数据的最小值。 • sum：表示取窗口内数据的总和。

示例：

- 查询分析：

```
* | select ts_smooth_simple(stamp, value) from ( select __time__ - __time__ % 120 as stamp, avg(v) as value from log GROUP BY stamp order by stamp )
```

- 输出结果：



显示项如下：

显示项		说明
横轴	unixtime	数据的Unixtime时间戳，单位为秒。
纵轴	src	未滤波前的数据。
	filter	滤波之后的数据。

7.9.3 多周期估计函数

多周期估计函数支持对不同时间段内的时序进行周期估计，利用傅立叶变换等一系列操作进行周期的提取。

ts_period_detect

该函数用于对时序数据进行分段周期估计。

函数格式：

```
select
 ts_period_detect(x, y,minPeriod,maxPeriod,samplePeriod,sampleMethod)
```

参数说明如下：

参数	说明	取值
x	时间列，从小到大排列。	格式为Unixtime时间戳，单位为秒。
y	数值列，对应某时刻的数据。	-

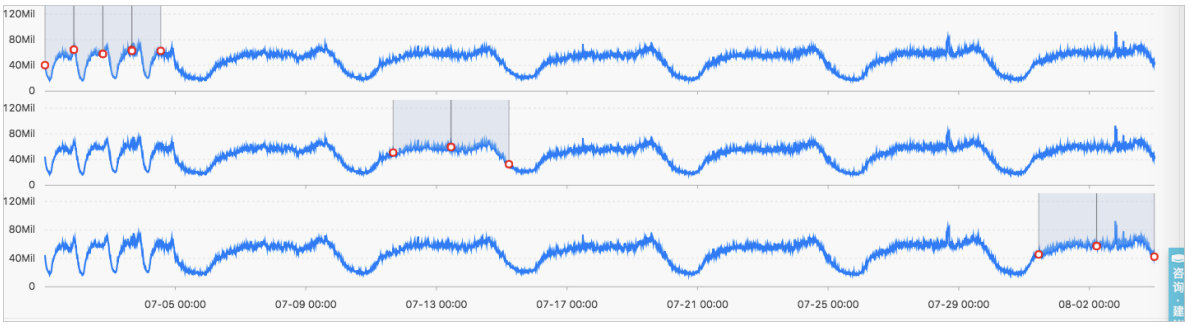
参数	说明	取值
minPeriod	预估计周期最小长度占序列总长度的比例。	小数形式，取值范围为(0,1]。
maxPeriod	预估计周期最大长度占序列总长度的比例。 说明： 指定参数时，*maxPeriod*必须大于*minPeriod*。	小数形式，取值范围为(0,1]。
samplePeriod	对当前时序数据进行采样的周期。	long类型，单位为秒。取值范围为1~86399。
sampleMethod	针对采样窗口内数据的采样方法。	取值包括： • avg：表示取窗口内数据的平均值。 • max：表示取窗口内数据的最大值。 • min：表示取窗口内数据的最小值。 • sum：表示取窗口内数据的总和。

示例：

- 查询分析：

```
* | select ts_period_detect(stamp, value, 0.2, 1.0, 1, 'avg') from
 ( select __time__ - __time__ % 120 as stamp, avg(v) as value from
log GROUP BY stamp order by stamp )
```

- 输出结果：



显示项如下：

显示项	说明
period_id	周期编号的数组，长度为1，其中值为0.0时，表示原始序列。
time_series	表示时间戳序列。
data_series	表示每个时间戳对应的结果。 • 当period_id为0.0时，表示原始序列结果。 • 当period_id不为0.0时，表示滤波之后的周期估计结果。

7.9.4 变点检测函数

变点检测函数一般用于对时序数据中的变点检测。

变点检测函数支持两种变点形态：

- 在指定时间段内的某些统计特性发生了变化。
- 序列中存在较为明显的断层。

函数列表

函数	说明
ts_cp_detect	寻找时序序列中具有不同统计特性的区间，区间端点即为变点。
ts_breakout_detect	寻找时序序列中，某统计量发生陡升或陡降的点。

ts_cp_detect

函数格式：

- 若您无法确定窗口大小，可以使用以下格式的ts_cp_detect函数，该函数调用的算法会默认使用长度等于10的窗口进行检测：

```
select ts_cp_detect(x, y, amplePeriod,sampleMethod)
```

- 若您需要根据业务曲线进行效果调试，可以使用以下格式的ts_cp_detect函数，通过设置参数minSize进行效果调优：

```
select ts_cp_detect(x, y, minSize, samplePeriod, sampleMethod)
```

参数说明如下：

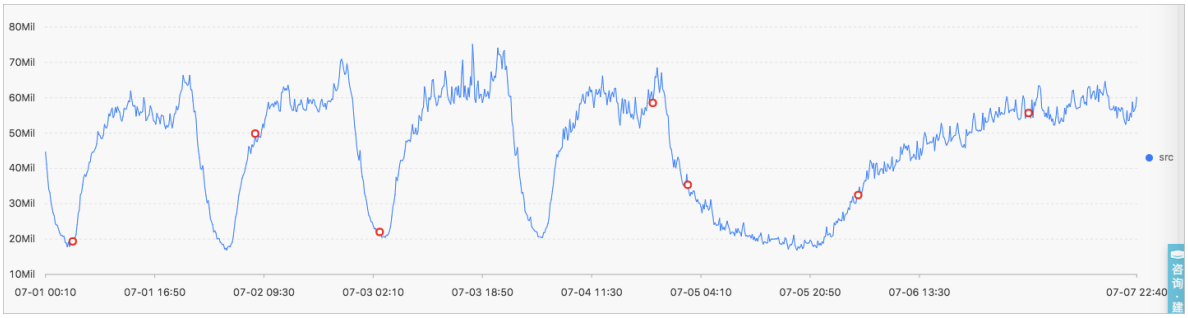
参数	说明	取值
x	时间列，从小到大排列。	格式为Unixtime时间戳，单位为秒。
y	数值列，对应某时刻的数据。	-
minSize	最小连续区间长度。	最小值为3，最大值不超过当前输入数据长度的1/10。
samplePeriod	对当前时序数据进行采样的周期。	long类型，取值范围为1~86399。
sampleMethod	针对采样窗口内数据的采样方法。	取值包括： • avg：表示取窗口内数据的平均值。 • max：表示取窗口内数据的最大值。 • min：表示取窗口内数据的最小值。 • sum：表示取窗口内数据的总和。

示例：

- 查询分析：

```
* | select ts_cp_detect(stamp, value, 3, 1, 'avg') from (select
__time__ - __time__ % 10 as stamp, avg(v) as value from log GROUP BY
 stamp order by stamp)
```

- 输出结果：



显示项如下：

显示项		说明
横轴	unixtime	数据的时间戳，单位为秒，例如1537071480。
纵轴	src	未滤波前的数据，例如1956092.7647745228。
	prob	该点为变点的概率，值的范围为0~1。

ts_breakout_detect

函数格式：

```
select ts_breakout_detect(x, y, winSize,  samplePeriod, sampleMethod)
```

参数说明如下：

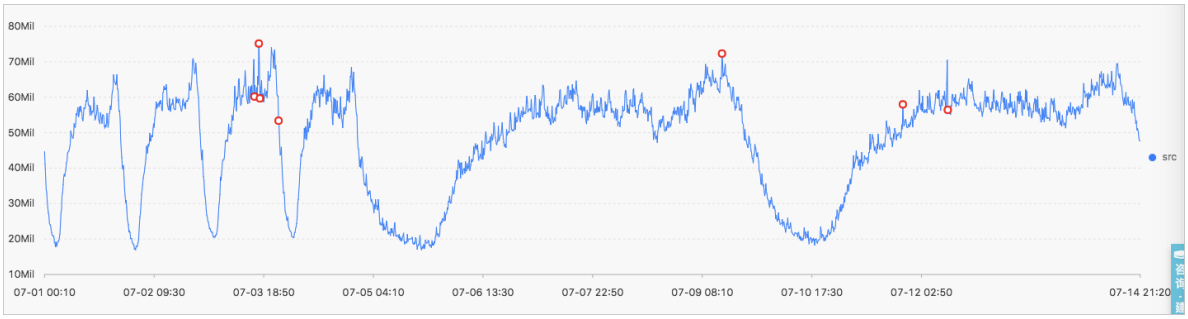
参数	说明	取值
x	时间列，从小到大排列。	格式为Unixtime时间戳，单位为秒。
y	数值列，对应某时刻的数据。	-
winSize	最小连续区间长度。	最小值为3，最大值不超过当前输入数据长度的1/10。
samplePeriod	对当前时序数据进行采样的周期。	long类型，单位为秒。取值范围为1~86399。
sampleMethod	针对采样窗口内数据的采样方法。	取值包括： • avg：表示取窗口内数据的平均值。 • max：表示取窗口内数据的最大值。 • min：表示取窗口内数据的最小值。 • sum：表示取窗口内数据的总和。

示例：

- 查询分析：

```
* | select ts_breakout_detect(stamp, value, 3, 1, 'avg') from (
select __time__ - __time__ % 10 as stamp, avg(v) as value from log
GROUP BY stamp order by stamp)
```

- 输出结果：



显示项如下：

显示项		说明
横轴	unixtime	数据的时间戳，单位为秒，例如1537071480。
纵轴	src	未滤波前的数据，例如1956092.7647745228。
	prob	该点为变点的概率，值的范围为0~1。

7.9.5 预测与异常检测函数

预测与异常检测函数通过预测时序曲线、寻找预测曲线和实际曲线之间误差的KSigma与分位数等特性进行异常检测。

函数列表

函数	说明
ts_predicate_simple	利用默认参数对时序数据进行建模，并进行简单的时序预测和异常点的检测。
ts_predicate_ar	使用自回归模型对时序数据进行建模，并进行简单的时序预测和异常点的检测。
ts_predicate_arma	使用移动自回归模型对时序数据进行建模，并进行简单的时序预测和异常点检测。
ts_predicate_arima	使用带有差分的移动自回归模型对时序数据进行建模，并进行简单的时序预测和异常点检测。



说明：

预测与异常检测系列函数的显示项相同，输出结果和对应说明请查看*ts_predicate_simple*输出结果示例。

ts_predicate_simple

函数格式：

```
select ts_predicate_simple(x, y, nPred, samplePeriod, sampleMethod)
```

参数说明如下：

参数	说明	取值
x	时间列，从小到大排列。	格式为Unixtime时间戳，单位为秒。

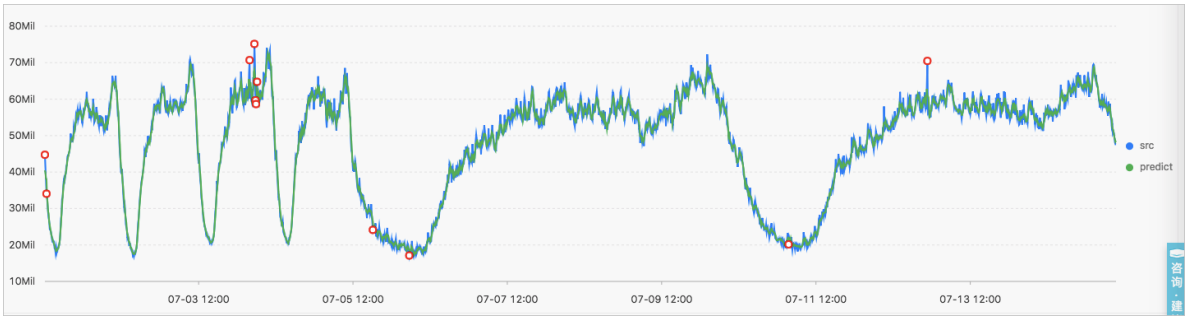
参数	说明	取值
y	数值列，对应某时刻的数据。	-
nPred	预测未来的点的数量。	long类型，取值范围为1~5*p。
samplePeriod	对当前时序数据进行采样的周期。	long类型，取值范围为1~86399。
sampleMethod	针对采样窗口内数据的采样方法。	取值包括： • avg：表示取窗口内数据的平均值。 • max：表示取窗口内数据的最大值。 • min：表示取窗口内数据的最小值。 • sum：表示取窗口内数据的总和。

示例：

- 查询分析：

```
* | select ts_predicate_simple(stamp, value, 6, 1, 'avg') from (
select __time__ - __time__ % 60 as stamp, avg(v) as value from log
GROUP BY stamp order by stamp)
```

- 输出结果：



显示项如下：

显示项		说明
横轴	unixtime	数据的Unixtime时间戳，单位为秒。
纵轴	src	原始数据。
	predict	滤波之后的数据。
	upper	预测的上界。当前置信度为默认为0.85，且不可修改。

显示项		说明
	lower	预测的下界。当前置信度为默认为0.85，且不可修改。
	anomaly_prob	该点是否为异常点的概率，值的范围为0~1。

ts_predicate_ar

函数格式：

```
select ts_predicate_ar(x, y, p, nPred, samplePeriod, sampleMethod)
```

参数说明如下：

参数	说明	取值
x	时间列，从小到大排列。	格式为Unixtime时间戳，单位为秒。
y	数值列，对应某时刻的数据。	-
p	自回归模型的阶数。	long类型，取值范围为2~8。
nPred	预测未来的点的数量。	long类型，取值范围为1~5**p*。
samplePeriod	对当前时序数据进行采样的周期。	long类型，取值范围为1~86399。
sampleMethod	针对采样窗口内数据的采样方法。	取值包括： • avg：表示取窗口内数据的平均值。 • max：表示取窗口内数据的最大值。 • min：表示取窗口内数据的最小值。 • sum：表示取窗口内数据的总和。

查询分析示例：

```
* | select ts_predicate_ar(stamp, value, 3, 4, 1, 'avg') from (select
__time__ - __time__ % 60 as stamp, avg(v) as value from log GROUP BY
stamp order by stamp)
```

ts_predicate_arma

函数格式：

```
select
 ts_predicate_arma(x, y, p, q, nPred, samplePeriod, sampleMethod)
```

参数说明如下：

参数	说明	取值
x	时间列，从小到大排列。	格式为Unixtime时间戳，单位为秒。
y	数值列，对应某时刻的数据。	-
p	自回归模型的阶数。	long类型，取值范围为2~8。
q	移动平均模型的阶数。	long类型，取值范围为2~8。
nPred	预测未来的点的数量。	long类型，取值范围为1~5**p*。
samplePeriod	对当前时序数据进行采样的周期。	long类型，取值范围为1~86399。
sampleMethod	针对采样窗口内数据的采样方法。	取值包括： • avg：表示取窗口内数据的平均值。 • max：表示取窗口内数据的最大值。 • min：表示取窗口内数据的最小值。 • sum：表示取窗口内数据的总和。

查询分析示例：

```
* | select ts_predicate_arma(stamp, value, 3, 2, 4, 1, 'avg') from (
select __time__ - __time__ % 60 as stamp, avg(v) as value from log
GROUP BY stamp order by stamp)
```

ts_predicate_arima

函数格式：

```
select
 ts_predicate_arima(x, y, p, d , q nPred, samplePeriod, sampleMethod)
```

参数说明如下：

参数	说明	取值
x	时间列，从小到大排列。	格式为Unixtime时间戳，单位为秒。
y	数值列，对应某时刻的数据。	-
p	自回归模型的阶数。	long类型，取值范围为2~8。
d	差分模型的阶数。	long类型，取值范围为1~3。
q	移动平均模型的阶数。	long类型，取值范围为2~8。
nPred	预测未来的点的数量。	long类型，取值范围为1~5**p*。
samplePeriod	对当前时序数据进行采样的周期。	long类型，取值范围为1~86399。
sampleMethod	针对采样窗口内数据的采样方法。	取值包括： • avg：表示取窗口内数据的平均值。 • max：表示取窗口内数据的最大值。 • min：表示取窗口内数据的最小值。 • sum：表示取窗口内数据的总和。

查询分析示例：

```
* | select ts_predicate_arima(stamp, value, 3, 1, 2, 4, 1, 'avg') from
 (select __time__ - __time__ % 60 as stamp, avg(v) as value from log
GROUP BY stamp order by stamp)
```

7.9.6 序列分解函数

序列分解函数提供针对业务曲线的分解功能，突出曲线的趋势信息和周期信息。

ts_decompose

函数格式：

```
select ts_decompose(x, y, samplePeriod, sampleMethod)
```

参数说明如下：

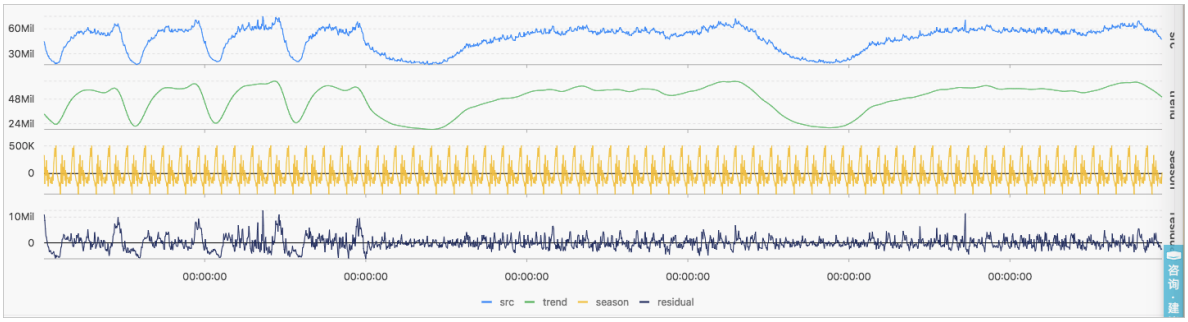
参数	说明	取值
x	时间列，从小到大排列。	格式为Unixtime时间戳，单位为秒。
y	数值列，对应某时刻的数据。	-
samplePeriod	对当前时序数据进行采样的周期。	long类型，取值范围为1~86399。
sampleMethod	针对采样窗口内数据的采样方法。	取值包括： • avg：表示取窗口内数据的平均值。 • max：表示取窗口内数据的最大值。 • min：表示取窗口内数据的最小值。 • sum：表示取窗口内数据的总和。

示例：

- 查询分析：

  ```
  * | select ts_decompose(stamp, value, 1, 'avg') from (select
  __time__ - __time__ % 60 as stamp, avg(v) as value from log GROUP BY
   stamp order by stamp)
  ```

- 输出结果：



显示项如下：

显示项		说明
横轴	unixtime	数据的Unixtime时间戳，单位为秒。
纵轴	src	原始数据。
	trend	分解出来的趋势数据。
	season	分解出来的周期数据。
	residual	分解出来的残差数据。

7.9.7 时序聚类函数

时序聚类函数针对输入的多条时序数据进行聚类，自动聚类出不同的曲线形态，进而快速找到相应的聚类中心和异于聚类中的其它形态曲线。

函数列表

函数	说明
ts_dencity_cluster	使用密度聚类方法对多条时序数据进行聚类。
ts_hierarchical_cluster	使用层次聚类方法对多条时序数据进行聚类。
ts_similar_instance	查找到指定曲线名称的相似曲线。

ts_dencity_cluster

函数格式：

```
select ts_dencity_cluster(x, y, z)
```

参数说明如下：

参数	说明	取值
x	时间列，从小到大排列。	格式为Unixtime时间戳，单位为秒。

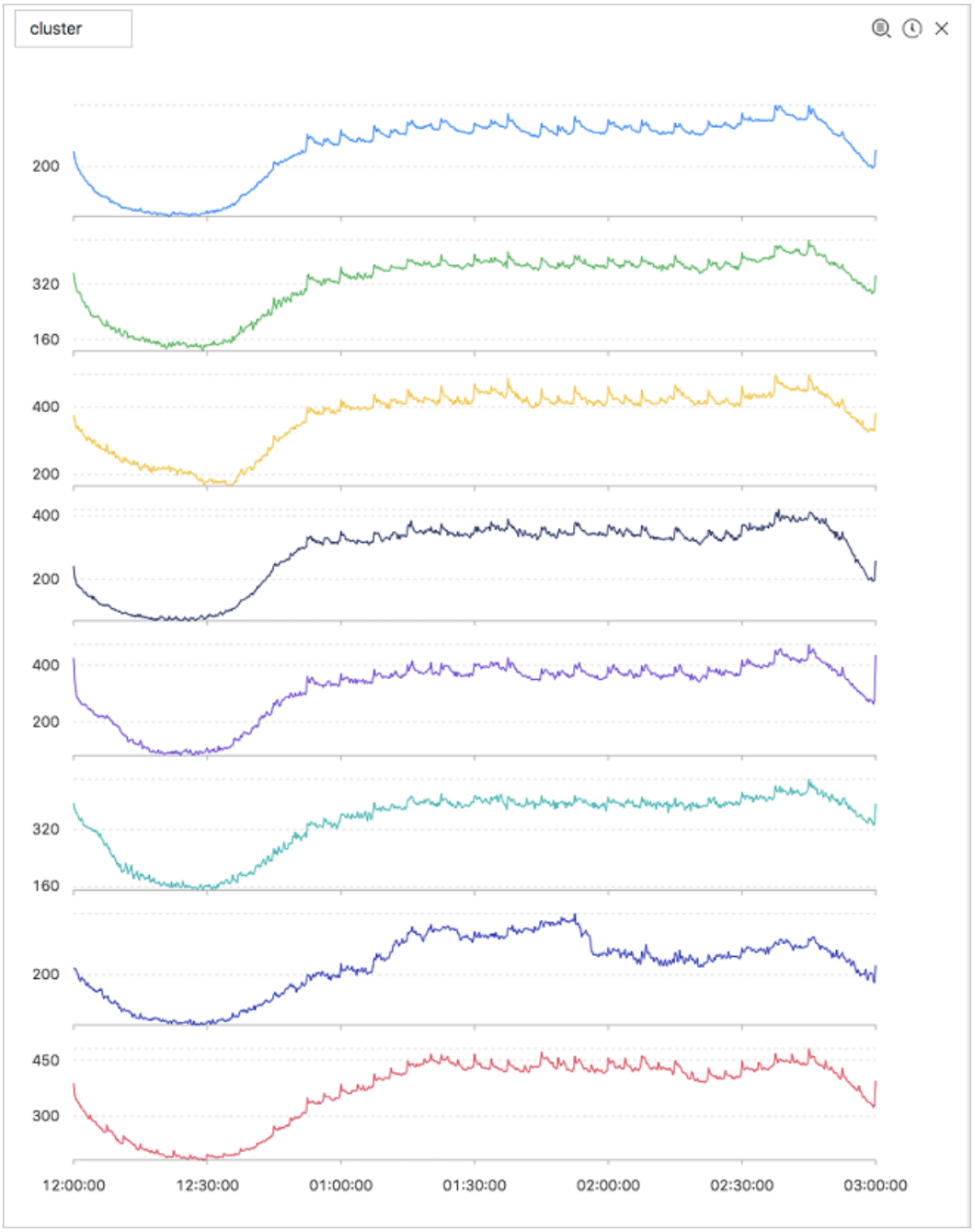
参数	说明	取值
y	数值列，对应某时刻的数据。	-
z	每个时刻数据对应的指标名称。	字符串类型，例如machine01.cpu_usr。

示例：

- 查询分析：

```
* and (h: "machine_01" OR h: "machine_02" OR h : "machine_03") |
select ts_dencity_cluster(stamp, metric_value,metric_name ) from (
select __time__ - __time__ % 600 as stamp, avg(v) as metric_value
, h as metric_name from log GROUP BY stamp, metric_name order BY
metric_name, stamp )
```

- 输出结果：



显示项如下：

显示项	说明
cluster_id	聚类的类别，其中-1表示未能划分到某一聚类中心。
rate	该聚类中的instance占比。
time_series	该聚类中心的时间戳序列。
data_series	该聚类中心的数据序列。
instance_names	该聚类中心包含的instance的集合。

显示项	说明
sim_instance	该类中的某一个instance名称。

ts_hierarchical_cluster

函数格式：

```
select ts_hierarchical_cluster(x, y, z)
```

参数说明如下：

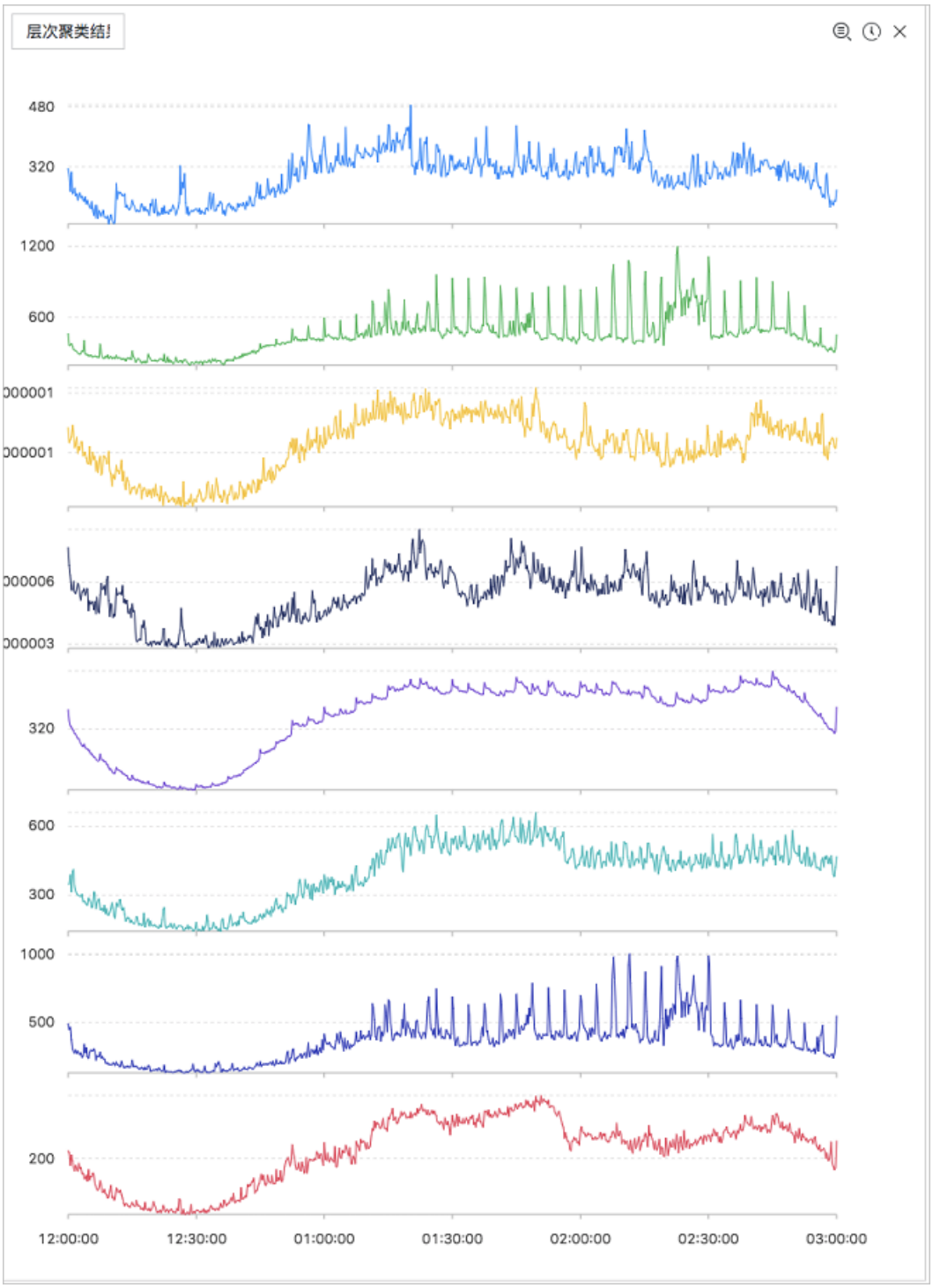
参数	说明	取值
x	时间列，从小到大排列。	格式为Unixtime时间戳，单位为秒。
y	数值列，对应某时刻的数据。	-
z	每个时刻数据对应的指标名称。	字符串类型，例如machine01.cpu_usr。

示例：

- 查询分析：

```
* and (h: "machine_01" OR h: "machine_02" OR h : "machine_03") |
select ts_hierarchical_cluster(stamp, metric_value, metric_name
) from ( select __time__ - __time__ % 600 as stamp, avg(v) as
metric_value, h as metric_name from log GROUP BY stamp, metric_name
order BY metric_name, stamp )
```

- 输出结果：



显示项如下：

显示项	说明
cluster_id	聚类的类别，其中-1表示未能划分到某一聚类中心。
rate	该聚类中的instance占比。
time_series	该聚类中心的时间戳序列。

显示项	说明
data_series	该聚类中心的数据序列。
instance_names	该聚类中心包含的instance的集合。
sim_instance	该类中的某一个instance名称。

ts_similar_instance

函数格式：

```
select ts_similar_instance(x, y, z , instance_name)
```

参数说明如下：

参数	说明	取值
x	时间列，从小到大排列。	格式为Unixtime时间戳，单位为秒。
y	数值列，对应某时刻的数据。	-
z	每个时刻数据对应的指标名称。	字符串类型，例如machine01.cpu_usr。
instance_name	指定某个待查找的指标的名字。	*z*集合中某个指标名称，字符串类型，如：machine01.cpu_usr。  说明： 必须是已创建的指标。

查询分析示例：

```
* and (h: "machine_01" OR h: "machine_02" OR h : "machine_03")
 | select ts_similar_instance(stamp, metric_value, metric_name, '
nu4e01524.nu8' ) from ( select __time__ - __time__ % 600 as stamp,
avg(v) as metric_value, h as metric_name from log GROUP BY stamp,
metric_name order BY metric_name, stamp )
```

显示项如下：

显示项	说明
instance_name	与指定指标相近的结果列表。
time_series	该聚类中心的时间戳序列。
data_series	该聚类中心的数据序列。

7.9.8 频繁模式统计函数

频繁模式统计函数可以在给定的多属性字段样本中，挖掘出具有一定代表性的属性组合，用来归纳对当前日志。

pattern_stat

函数格式：

```
select pattern_stat(array[col1, col2, col3], array['col1_name',
 'col2_name', 'col3_name'], array[col5, col6], array['col5_name',
 'col6_name'], supportScore, sample_ratio)
```

参数说明如下：

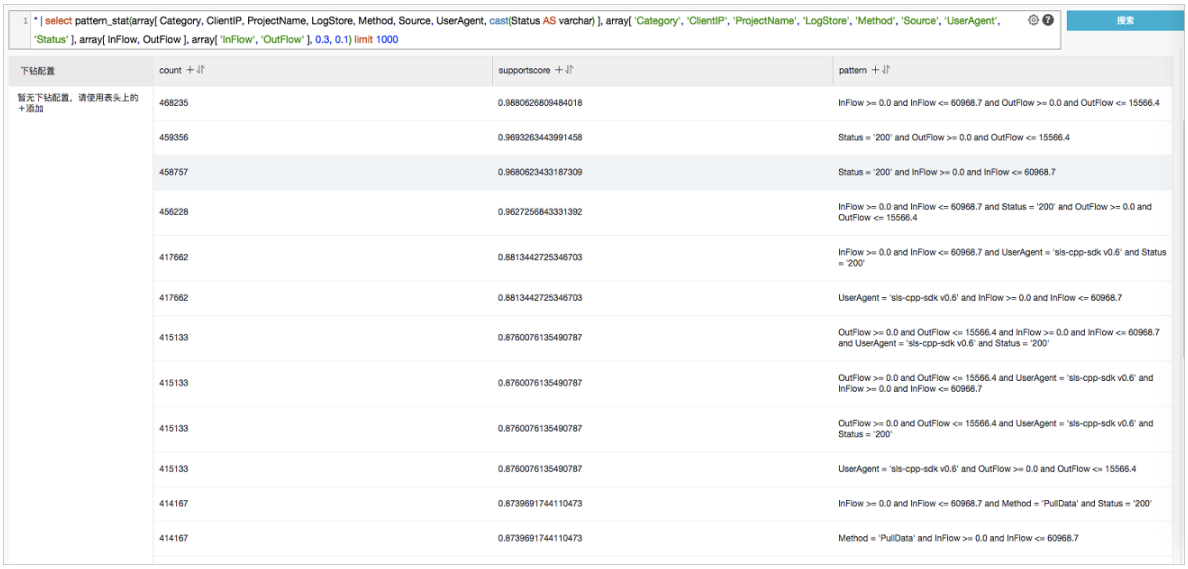
参数	说明	取值
array[col1, col2, col3]	字符型数据的输入列。	数组形式，例如：array[clientIP, sourceIP, path, logstore]。
array['col1_name', 'col2_name', 'col3_name']	字符型数据的输入列的对应名称。	数组形式，例如：array['clientIP', 'sourceIP', 'path', 'logstore']。
array[col5, col6]	数值型数据的输入列。	数组形式，例如：array[Inflow, OutFlow]。
array['col5_name', 'col6_name']	数值型数据的输入列的对应名称。	数组形式，例如array['Inflow', 'OutFlow']。
supportScore	正负样本在进行模式挖掘时的支持度。	double类型，取值为(0,1]。
sample_ratio	采样比率，默认为0.1，表示只拿10%全量集合。	double类型，取值为(0,1]。

示例：

- 查询分析：

```
* | select pattern_stat(array[ Category, ClientIP, ProjectName,
LogStore, Method, Source, UserAgent ], array[ 'Category', 'ClientIP
', 'ProjectName', 'LogStore', 'Method', 'Source', 'UserAgent' ],
array[ InFlow, OutFlow ], array[ 'InFlow', 'OutFlow' ], 0.45, 0.3)
limit 1000
```

- 输出结果：



显示项如下：

显示项	说明
count	当前模式所含样本的数量。
supportScore	当前模式的支持度。
pattern	模式的具体内容，按照条件查询的形式组织。

7.9.9 差异模式统计函数

差异模式统计函数基于给定的多属性字段样本，在给定的判别条件下，分析出影响该条件划分的差异化模式集合，帮助您快速诊断导致当前判别条件差异的原因。

pattern_diff

函数格式：

```
select
 pattern_diff(array_char_value, array_char_name, array_numeric_value, array_numer
 )
```

参数说明如下：

参数	说明	取值
array_char_value	字符型数据的输入列。	数组形式，例如：array[clientIP, sourceIP, path, logstore]。
array_char_name	字符型数据的输入列的对应名称。	数组形式，例如：array['clientIP', 'sourceIP', 'path', 'logstore']。

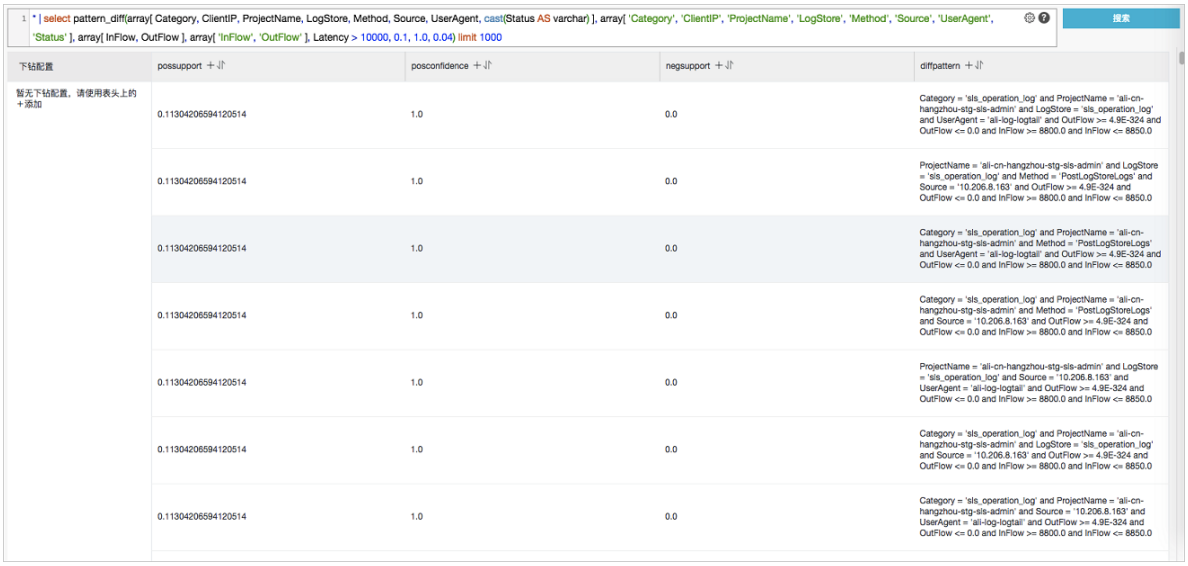
参数	说明	取值
array_numeric_value	数值型数据的输入列。	数组形式，例如：array[Inflow, OutFlow]。
array_numeric_name	数值型数据的输入列的对应名称。	数组形式，例如array['Inflow', 'OutFlow']。
condition	筛选数据的条件。条件为True则为正样本，条件为False则为负样本。	例如：Latency <= 300。
supportScore	正负样本在进行模式挖掘时的支持度。	double类型，取值为(0,1]。
posSampleRatio	正样本的采样率。默认为0.5，表示只取50%正样本集合。	double类型，取值为(0,1]。
negSampleRatio	负样本的采样率，默认为0.5，表示只取50%负样本集合。	double类型，取值为(0,1]。

示例：

- 查询分析：

```
* | select pattern_diff(array[ Category, ClientIP, ProjectName, LogStore, Method, Source, UserAgent ], array[ 'Category', 'ClientIP', 'ProjectName', 'LogStore', 'Method', 'Source', 'UserAgent' ], array[ InFlow, OutFlow ], array[ 'InFlow', 'OutFlow' ], Latency > 300, 0.2, 0.1, 1.0) limit 1000
```

- 输出结果：



显示项如下：

显示项	说明
possupport	挖掘出来的模式在正样本中的支持度。
posconfidence	挖掘出来的模式在曾样本中的置信度。
negsupport	挖掘出来的模式在负样本中的支持度。
diffpattern	挖掘出来的具体模式内容。

7.10 分析进阶

7.10.1 优秀分析案例

案例列表

1. 5分钟错误率超过40%时触发报警
2. 当流量暴跌时#触发报警
3. 按照数据区间分桶#在每个桶内计算平均延时
4. 在group by的结果中#返回百分比
5. 统计满足条件的个数

5分钟错误率超过40%时触发报警

统计每分钟的500错误率，当最近5分钟错误率超过40%时触发报警。

```
status:500 | select __topic__, max_by(error_count,window_time)/1.0/sum
(error_count) as error_ratio, sum(error_count) as total_error  from (
select __topic__, count(*) as error_count , __time__ - __time__ % 300
  as window_time  from log group by __topic__, window_time
)
```

```
group by __topic__  having  max_by(error_count,window_time)/1.0/sum(
error_count)   > 0.4  and sum(error_count) > 500 order by total_error
desc limit 100
```

当流量暴跌时，触发报警

统计每分钟的流量，挡最近的流量出现暴跌时，触发报警。 由于在最近的一分钟内，统计的数据不是一个完整分钟的，所以，需要除以(max(**time**) - min(**time**)) 进行归一化，统计每个分钟内的流量均值。

```
* | SELECT SUM(inflow) / (max(__time__) - min(__time__))  as
inflow_per_minute, date_trunc('minute',__time__)  as minute group by
minute
```

按照数据区间分桶，在每个桶内计算平均延时

```
* | select avg(latency) as latency , case when originSize < 5000 then
 's1' when originSize < 20000 then 's2' when originSize < 500000 then
 's3' when originSize < 100000000 then 's4' else 's5' end as os group
by os
```

在group by的结果中，返回百分比

不同部门的count结果，及其所占百分比。该query结合了子查询、窗口函数。其中sum(c) over() 表示计算所有行的和。

```
* |  select   department, c*1.0/ sum(c) over ()  from(select  count(1
) as c, department   from log groupby department)
```

统计满足条件的个数

在URL路径中，我们需要根据URL不同的特征，来计数，这种情况，可以使用CASE WHEN语法，但还有个更简单的语法是count_if。

```
* | select  count_if(uri like '%login') as login_num, count_if(uri
like '%register') as register_num, date_format(date_trunc('minute',
__time__), '%m-%d %H:%i') as time  group by time order by time limit
100
```

7.10.2 优化查询

不同的query在分析时的效率是不同的，在此为您提供部分常见的优化query方式，供您参考。

尽量避免对字符串列进行GROUP BY计算

对字符串进行GROUP BY，会导致大量的hash计算，这部分计算量往往会占据整体计算的50%以上。

例如以下两个query：

```
* | select count(1) as pv , date_trunc('hour',__time__) as time group
by time
* | select count(1) as pv , from_unixtime(__time__-__time__%3600) as
time group by __time__-__time__%3600
```

Query 1 和2达到的效果是相同的，都是计算每个小时的日志count数，但是Query 1 首先把时间转化成字符串，例如`2017-12-12 00:00:00`， 然后对这个字符串进行GROUP BY。 Query 2是先对时间整点值进行计算，GROUP BY计算后才会转化成字符串类型。Query 1需要对字符串进行hash操作，所以在执行效率上，Query 2更佳。

GROUP BY多列时，把字典大的字段放在前面

例如，province有13个，用户有1亿。

```
快：  * | select province,uid,count(1)groupby province,uid
慢：   * | select province,uid,count(1)groupby uid,province
```

使用估算函数

估算函数的性能要比精确计算好很多。估算会损失一些可接受的精确度，来达到快速计算的效果。

```
快： * |select approx_distinct(ip)
慢：  * | select count(distinct(ip))
```

在SQL中获取需要的列，尽量不要读取所有列

获取所有列，请使用查询语法。在SQL计算时，尽量只读取需要参与计算的列，这会加快计算。

```
快 : * |select a,b c
慢 :* |select*
```

非group by的列，尽量放到聚合函数中

例如，userid，用户名，必定是一一对应的，我们只需要按照userid进行group by即可。

```
快： * | select userid, arbitrary(username), count(1)groupby userid
慢：  * | select userid, username, count(1)groupby userid,username
```

7.10.3 查询分析全方位对比（ELK）

提到日志实时分析，很多人都会想到很火的ELK Stack（Elastic/Logstash/Kibana）来搭建。ELK方案开源，在社区中有大量的内容和使用案例。

阿里云日志服务产品在新版中增强查询分析功能（LogSearch/Analytics），支持对日志数据实时索引与查询分析，并且对查询性能和计算数据量做了大量优化。在这里我们做一个全方位的比较，对于用户关心的点，我们依次展开分析：

- 易用：上手及使用过程中的代价
- 功能（重点）：主要针对查询与分析两块
- 性能（重点）：对于单位大小数据量查询与分析需求，延时如何
- 规模（重点）：能够承担的数据量，扩展性等
- 成本（重点）：同样功能和性能，使用分别花多少钱

易用

对日志分析系统而言，有如下使用过程：

1. 采集：将数据稳定写入
2. 配置：如何配置数据源
3. 扩容：接入更多数据源，更多机器，对存储空间，机器进行扩容
4. 使用：这部分在功能这一节介绍
5. 导出：数据能否方便导出到其他系统，例如做流计算、放到对象存储中进行备份
6. 多租户：如何将数据分享给他人使用，使用是否安全等

以下是比较结果：

项目	小项	自建**ELK**	**Log Analytics**
采集	协议	Restful API	Restful API
	Agent	Logstash/Beats/ FluentD，生态十分丰富	Logtail（为主）+其他（例如Logstash）
配置	单元	提供Index概念用以区分不同日志	Project + Logstore。提供两层概念，Project相当于命名空间，可以在Project下建立多个Logstore
	属性	API + Kibana	API + SDK + 控制台
扩容	存储	增加机器，购买云盘	无需操作
	机器	新增机器	无需操作

项目	小项	自建ELK	Log Analytics
	配置	配置Logstash并通过配管系统应用机器	控制台/API 操作，无需配管系统
	采集点	通过配管系统控制，将配置和Logstash安装到机器组	控制台/API 操作，无需配管系统
导出	方式	API/SDK	API/SDK + 各流计算引擎（Spark，Storm，Flink，StreamCompute，CloudMonitor，ARMS） + 存储（OSS + MaxCompute）
多租户	安全	无	https + 传输签名 + 多租户隔离 + 访问控制
	使用	同一账号	子账号，角色，产品，临时授权等

总体而言：

ELK 有非常多的生态和写入工具，使用中的安装、配置等都有较多工具可以参考。LogSearch/Analytics是一个托管服务，从接入、配置、使用上集成度非常高，普通用户5分钟就可以接入。并且过程中不需要担心容量、并发等问题，按量计费，弹性伸缩。

功能（查询+分析）

查询主要将符合条件的日志快速命中，分析功能是对数据进行统计与计算。

例如我们有如下需求：所有状态码大于200读请求，根据Ip统计次数和流量，这样的分析请求就可以转化为两个操作：查询到指定结果，对结果进行统计分析。在一些情况下我们也可以不进行查询，直接对所有日志进行分析。

```
1. Status in (200,500] and Method:Get*
 2. select count(1) as c, sum(inflow) as sum_inflow, ip group by Ip
```

查询能力对比

类型	小项	自建ELK	Log-Search/Analytics
文本	索引查询	支持	支持

类型	小项	自建ELK	Log-Search/ Analytics
	分词	支持	支持
	中文分词	支持	支持
	前缀	支持	支持
	后缀	支持	
	模糊	支持	支持
	Wildcast	支持	（计划支持）
数值	long	支持	支持
	double	支持	支持
Nested	Json查询	支持	
Geo	Geo查询	支持	不直接支持，但可以通过区间查询代替
Ip	Ip查询	支持	不直接支持，通过字符串支持
上下文	上下文查询		支持
	上下文过滤		支持

ElasticSearch 支持的数据类型，以及查询方式上要更强。Log-Search/Analytics 支持大部分常用查询，也有一些特色（例如程序日志上下文查询与展开）。

分析能力对比

- *ES 5.5 Aggregation*
- 实时分析简介

类型	小项	自建ELK	Log-Search/ Analytics
接口	方式	API/SDK	API/SDK + SQL92
	其他协议		JDBC
Agg	Bucketing	支持	支持
	Metric	支持	支持
	Matrix	支持	支持

类型	小项	自建ELK	Log-Search/Analytics
	Pipeline	有限支持	完整支持
运算	数值		支持
	字符串		支持
	估算		支持
	数理统计		支持
	日期转换		支持
GroupBy	Agg	支持	支持
	Having条件		支持
排序	排序		支持
Join	多表Join		支持

从结果看Log-Search/Analytics 提供的功能是ES 超集，完整支持SQL 92 语义，只要会写SQL 都能直接使用。

性能

接下来我们测试针对相同数据集，分别对比写入数据及查询，和聚合计算能力。

实验环境

1. 测试配置

类别	自建ELK	LogSearch/LogAnalytics
环境	ECS 4核16GB * 4台 + 高效云盘 SSD	-
Shard	10	10
拷贝数	2	3（默认配置，对用户不可见）

2. 测试数据

- 5列double
- 5列long
- 5列text，字典大小分别是256，512，768，1024，1280

以上字段完全随机，如下为一条测试日志样例：

```
timestamp:August 27th 2017, 21:50:19.000
 long_1:756,444 double_1:0 text_1:value_136
 long_2:-3,839,872,295 double_2:-11.13 text_2:value_475
 long_3:-73,775,372,011,896 double_3:-70,220.163 text_3:value_3
 long_4:173,468,492,344,196 double_4:35,123.978 text_4:value_124
 long_5:389,467,512,234,496 double_5:-20,10.312 text_5:value_1125
```

3. 数据集大小

- 原始数据大小：50 GB
- 原始数据除去Key后内容大小：27 GB（LogSearch/Analytics 以该大小作为存储计费单元）
- 日志行数：162,640,232 （约为1.6亿条）

写入测试结果

ES采用bulk api批量写入，LogSearch/Analytics 用PostLogstoreLogs API批量写入，结果如下：

类型	项目	自建**ELK**	**LogSearch/Analytics**
延时	平均写入延时	40 ms	14 ms
存储	单拷贝数据量	86G	58G
	膨胀率：数据量/原始数据大小	172%	121%

图 7-27: 平均写入延时对比

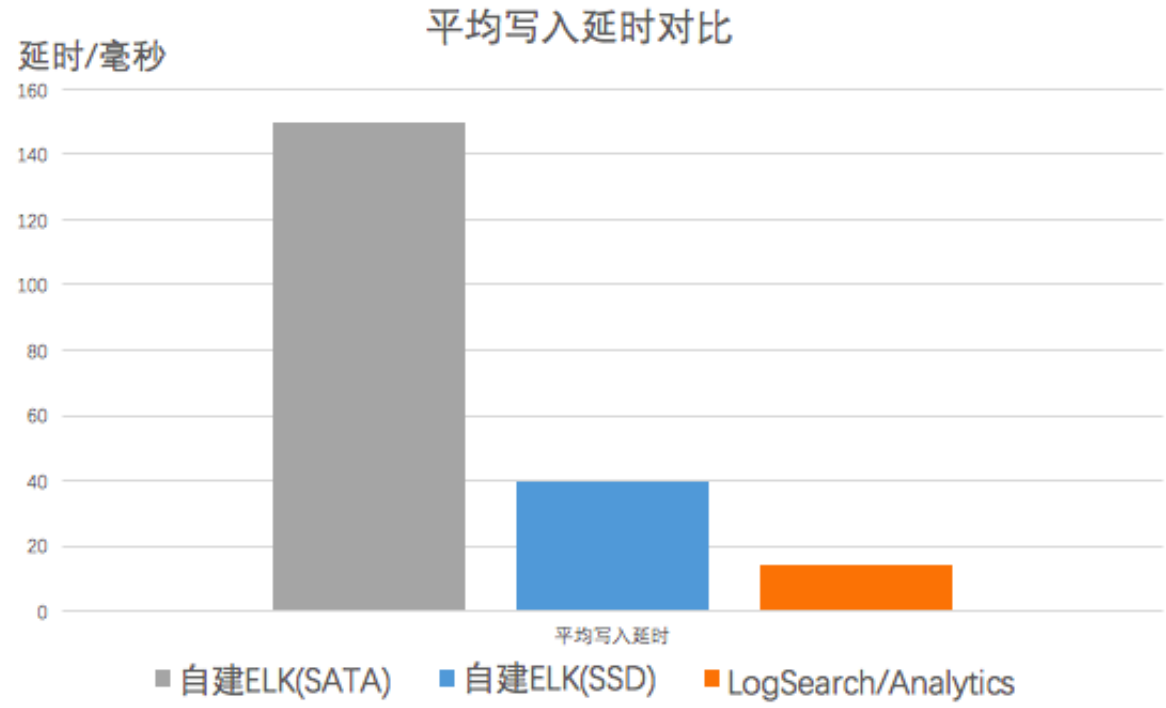
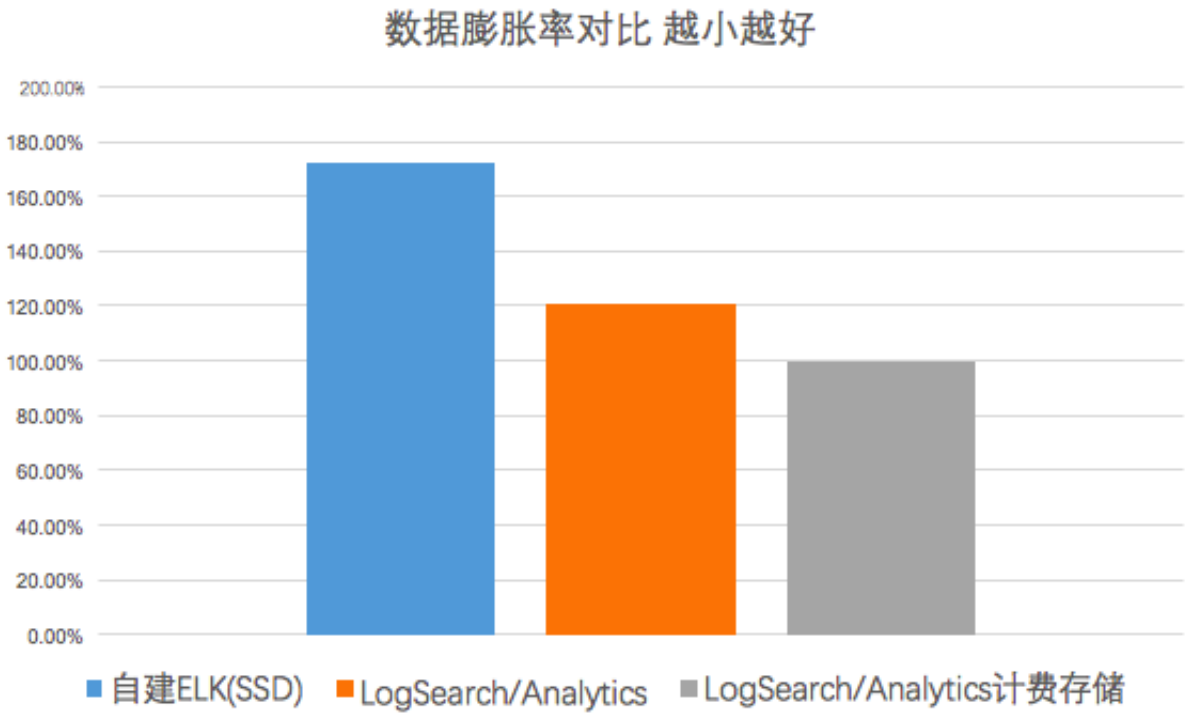


图 7-28: 数据膨胀率对比



说明：

LogSearch/Analytics 产生计费的存储量包括压缩的原始数据写入量（23G）+索引流量（27G），共50G存储费用。

从测试结果来看

- LogSearch/Analytics写入延时好于ES，14 ms vs 40ms
- 空间：原始数据50G，因测试数据比较随机所以存储空间会有膨胀（大部分真实场景下，存储压缩后会比原始数据小）。ES胀到86G，膨胀率为172%，在存储空间超出LogSearch/Analytics 58%

读取（查询+分析）测试

测试场景

我们在此选取两种比较常见的场景：日志查询和聚合计算。分别统计并发度为1，5，10时，两种case的平均延时。

1. 针对全量数据，对任意text列计算group by，计算5列数值的avg/min/max/sum/count，并按照count排序，取前1000个结果，例如：

```
select count(long_1) as pv,sum(long_2),min(long_3),max(long_4),sum(
long_5)
group by text_1 order by pv desc limit 1000
```

2. 针对全量数据，随机查询日志中的关键词，例如查询“value_126”，获取命中的日志数目与前100行，例如：

```
value_126
```

测试结果

类型	并发数	ES延时（单位s）	LogSearch/Analytics 延时（单位s）
case1：分析类	1	3.76	3.4
	5	3.9	4.7
	10	6.6	7.2
case2：查询类	1	0.097	0.086
	5	0.171	0.083
	10	0.2	0.082

结果分析

- 从结果看，对于1.5亿数据量这个规模，两者都达到了秒级查询与分析能力。
- 针对统计类场景（case 1），ES和日志服务延时处同一量级。ES采用SSD云盘，在读取大量数据时IO优势比较高。
- 针对查询类场景（case 2），LogAnalytics在延时明显优于ES。随着并发的增加，ELK延时对应增加，而LogAnalytics延时保持稳定甚至略有下降。

图 **7-29: case1** 分析类延时对比

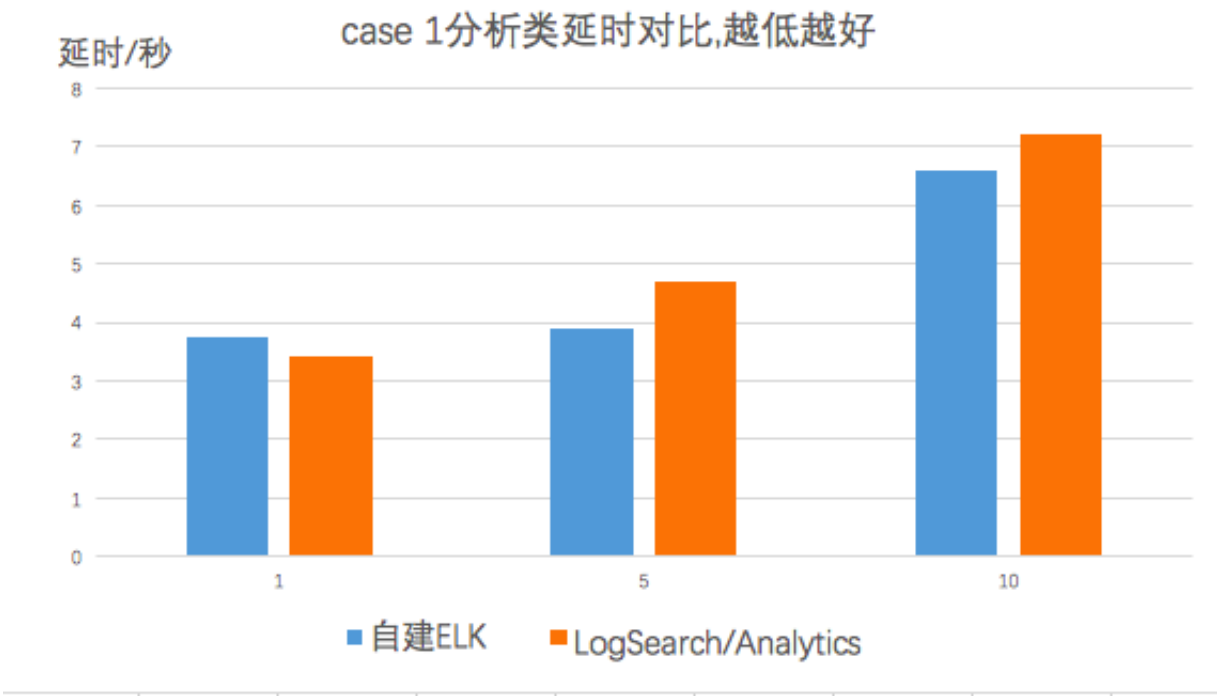
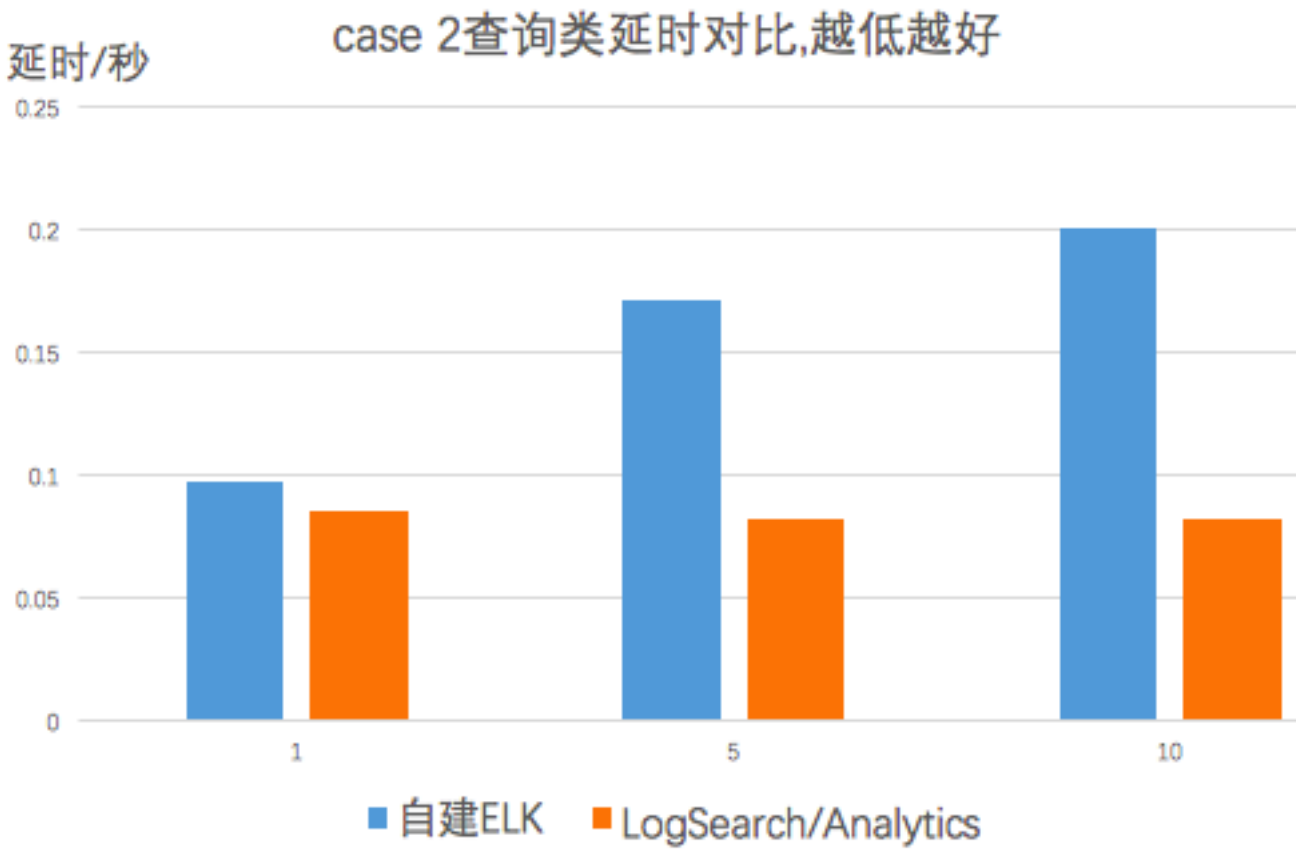


图 7-30: case2 查询类延时对比



规模

1. LogSearch/Analytics 一天可以索引PB级数据，一次查询可以在秒级过几十TB规模数据，在数据规模上可以做到弹性伸缩与水平扩展
2. ES比较适合服务场景为：写入GB-TB/Day、存储在TB级。主要受限于2个原因：
   - 单集群规模：比较理想为20台左右，业界比较大的为100节点一个集群
   - 写入扩容：shard创建后便不可再修改，当吞吐率增加时，需要动态扩容节点，最多可使用的节点数便是shard的个数
   - 存储扩容：主shard达到磁盘的上线时，要么迁移到更大的一块磁盘上，要么只能分配更多的shard。一般做法是创建一个新的索引，指定更多shard，并且rebuild旧的数据

   LogSearch/Analytics 则没有扩展性方面的问题，每个shard都是分布式存储。并且当吞吐率增加时，可以动态分裂shard，达到处理能力水平扩展。

成本

以上述测试数据为例，一天写入50GB数据（其中27GB 为实际的内容），保存90天，平均一个月的耗费。

1. 日志服务（LogSearch/LogAnalytics）计费规则参考计费方式，包括读写流量、索引流量、存储空间等计费项，查询功能免费。

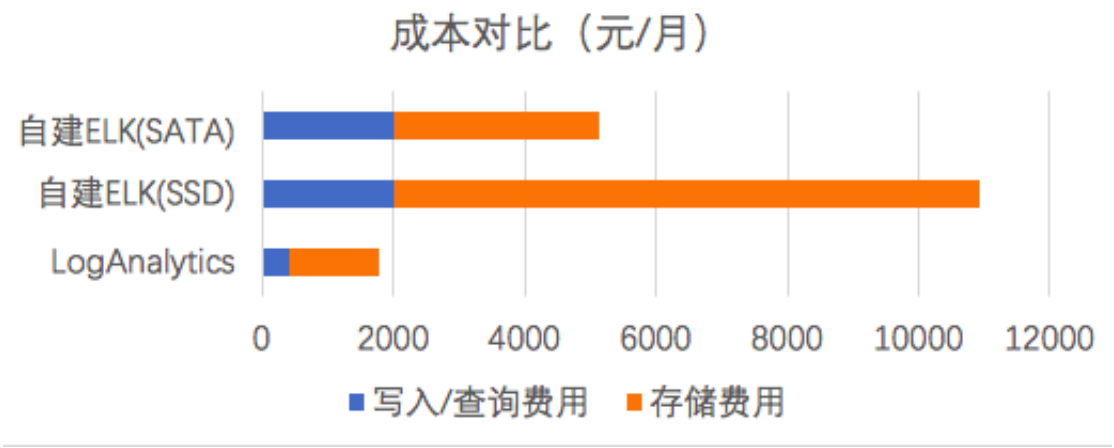
计费项目	值	单价	费用（元)
读写流量	23G * 30	0.2 元/GB	138
存储空间（保存90天）	50G * 90	0.3 元/GB*Month	1350
索引流量	27G * 30	0.35 元/GB	283
总计	-	-	1771

2. ES费用包括机器费用，及存储数据SSD云盘费用

   - 云盘一般可以提供高可靠性，因此我们这里不计费副本存储量。
   - 存储盘一般需要预留15%剩余空间，以防空间写满，因此乘以一个1.15系数。

计费项目	值	单价	费用（元)
服务器	4台4核16G（三个月）（ecs.mn4.xlarge）	包年包月费用：675 元/Month	2021
存储	86 * 1.15 * 90（这里只计算一个副本）	SSD：1 元/GB*M	8901
	-	SATA：0.35 元/GB*M	3115
总计			12943（SSD）
			5135（SATA）

图 **7-31:** 成本对比



同样性能，使用LogSearch/Analytics与ELK（SSD）费用比为 13.6%。在测试过程中，我们也尝试把SSD换成SATA以节省费用（LogAnalytics与SATA版费用比为 34%），但测试发现延时会从40ms上升至150ms，在长时间读写下，查询和读写延时变得很高，无法正常工作了。

总结

LogSearch/Analytics在提供同样查询速度、更高吞吐量、更强分析能力背后，只需要开源方案13%成本，并且按量付费免运维，让您把精力放在业务分析上。

日志服务除了LogSearch/Analytics外，还提供LogHub、LogShipper功能，支持实时采集数据、与流计算系统（Spark、Storm、Flink、StreamCompute）、离线分析系统（MaxCompute、E-MapReduce、Presto、Hive等）打通，提供一站式实时数据解决方案。

7.10.4 查询方案（ELK/Hive）对比

DevOps 场景日志查询方案对比：ELK（搜索类）、Hadoop/Hive

在互联网大潮中，为应对不断加速的软件、服务交付需求，无论是在创业团队还是大型互联网公司，都已经转向或逐步转向 DevOps 模式，通过开发（Dev）和运维支持（Ops）之间有效协作，解决跨部门协作、快速响应客户需求、进行持续交付。

日志在 DevOps 下显得越发重要，无论是在问题调查、安全审计、运营支撑等各方面，日志都起到了重要的支撑作用。一个合适的日志解决方案，对于 DevOps 显得尤为重要。

我们从如下方面考察 LogSearch 与 ELK、Hadoop/Hive 类方案的对比：

- 延时：日志产生后，多久可查询
- 查询能力：单位时间扫描数据量

- 查询功能：关键词查询、条件组合查询、模糊查询、数值比较、上下文查询
- 弹性：快速应对百倍流量上涨
- 成本：每 GB 费用
- 可靠性：日志数据安全不丢失

常用方案以及对比

- 自建 ELK：通过 Elastic、Logstash、Kibana 进行对比
- 离线 Hadoop + Hive：将数据存储在 Hadoop，利用 Hive 或 Presto 进行查询（非分析）
- 使用日志服务（LogSearch）

以应用程序日志和 Nginx 访问日志为例（每天 10GB），对比几种方案。

功能项	ELK 类系统	Hadoop + Hive	日志服务
可查延时	1~60 秒(由 refresh_interval 控制)	几分钟~数小时	实时
查询延时	小于 1 秒	分钟级	小于 1 秒
超大查询	几十秒~数分钟	分钟级	秒级（查询 10 亿日志）
关键词查询	支持	支持	支持
模糊查询	支持	支持	支持
上下文查询	不支持	不支持	支持
数值比较	支持	支持	支持
连续字符串查询	支持	支持	不支持
弹性	提前预备机器	提前预备机器	秒级 10 倍扩容
写入成本	写入5 元/GB，查询免费	写入免费，查询一次 0.3 元/GB	写入 0.5 元/GB，查询免费
存储成本	≤ 3.36 元/GB*天	≤ 0.035 元/GB*天	≤ 0.016 元/GB*天
可靠性	设置拷贝数	设置拷贝数	SLA > 99.9%，数据 > 99.99999999%

7.11 通过JDBC协议分析日志

除概览外，您还可以使用JDBC + 标准SQL 92进行日志查询与分析。

连接参数

连接参数	示例	说明
host	regionid.example.com	服务入口，目前仅支持经典网络内网访问和VPC网络访问
port	10005	默认使用10005作为端口号
user	bq2sjzesjmo86kq	访问秘钥 AccesskeyId
password	4fdO1fTDDuZP	访问秘钥Accesskey
database	sample-project	账号下的项目*#Project#*
table	sample-logstore	项目下的日志库*#Logstore#*

例如通过MySQL命令连接示例如下：

```
mysql -hcn-shanghai-intranet.log.aliyuncs.com -ubq2sjzesjmo86kq -
p4fdO1fTDDuZP -P10005
use sample-project; // 使用某个Project
```

前提条件

访问JDBC接口，必须使用主账号的AK或者子帐号的AK。子帐号必须是Project owner的子帐号，同时子帐号具有Project级别的读权限。

语法说明

注意事项

在where条件中必须包含`__date__`或`__time__`来限制查询的时间范围。`__date__`是timestamp类型`__time__`是bigint类型。

例如：

- `__date__ > '2017-08-07 00:00:00' and __date__ < '2017-08-08 00:00:00'`
- `__time__ > 1502691923 and __time__ < 1502692923`

上述两种条件必须出现一个。

过滤语法

关于where下过滤（filter）语法如下：

语义	示例	说明
字符串搜索	`key = "value"`	查询的是分词之后的结果。
字符串模糊搜索	`key = "valu*"`	查询的是分词之后模糊匹配的结果。
数值比较	`num_field > 1`	支持的比较运算符包括`>`、 `>=`、`=`、 `<`和`<=`。
逻辑运算	`and or not`	例如`a = "x" and b ="y"`或`a = "x" and not b ="y"`。
全文搜索	`__line__ ="abc"`	如果使用全文索引搜索，需要使用特殊的key（`__line__`）。

计算语法

支持计算操作符参见分析语法。

SQL92语法

过滤 + 计算组合为SQL92语法。

例如对于如下查询：

```
status>200 |select avg(latency),max(latency) ,count(1) as c GROUP BY
method  ORDER BY c DESC  LIMIT 20
```

我们可以将查询中过滤部分+ 时间条件组合成为查询的条件，变成标准SQL92语法：

```
select avg(latency),max(latency) ,count(1) as c from sample-logstore
 where status>200 and __time__>=1500975424 and __time__ < 1501035044
GROUP BY  method  ORDER BY c DESC  LIMIT 20
```

通过JDBC协议访问

程序调用

开发者可以在任何一个支持MySQL connector的程序中使用MySQL语法连接日志服务。例如使用JDBC或者Python MySQLdb。

使用样例：

```
import com.mysql.jdbc.*;
import java.sql.*;
import java.sql.Connection;
import java.sql.ResultSetMetaData;
import java.sql.Statement;
public class testjdbc {
```

```
    public static void main(String args[]){
        Connection conn = null;
        Statement stmt = null;
        try {
            //STEP 2: Register JDBC driver
            Class.forName("com.mysql.jdbc.Driver");
            //STEP 3: Open a connection
            System.out.println("Connecting to a selected database
...");
            conn = DriverManager.getConnection("jdbc:mysql://cn-
shanghai-intranet.log.aliyuncs.com:10005/sample-project","accessid","
accesskey");
            System.out.println("Connected database successfully...");
            //STEP 4: Execute a query
            System.out.println("Creating statement...");
            stmt = conn.createStatement();
            String sql = "SELECT method,min(latency,10)  as c,max
(latency,10) from sample-logstore where  __time__>=1500975424 and
__time__ < 1501035044 and latency > 0  and latency < 6142629 and  not
 (method='Postlogstorelogs' or method='GetLogtailConfig') group by
method " ;
            String sql-example2 = "select count(1) ,max(latency),
avg(latency), histogram(method),histogram(source),histogram(status),
histogram(clientip),histogram(__source__) from  test10 where __date__
  >'2017-07-20 00:00:00'  and  __date__ <'2017-08-02 00:00:00' and
__line__='abc#def' and latency < 100000 and (method = 'getlogstorelogS
' or method='Get**' and method <> 'GetCursorOrData' )";
            String sql-example3 = "select count(1) from  sample-
logstore where     __date__  >       '2017-08-07 00:00:00' and
__date__ <     '2017-08-08 00:00:00' limit 100";
            ResultSet rs = stmt.executeQuery(sql);
            //STEP 5: Extract data from result set
            while(rs.next()){
                //Retrieve by column name
                ResultSetMetaData data = rs.getMetaData();
                System.out.println(data.getColumnCount());
                for(int i = 0;i < data.getColumnCount();++i) {
                    String name = data.getColumnName(i+1);
                    System.out.print(name+":");
                    System.out.print(rs.getObject(name));
                }
                System.out.println();
            }
            rs.close();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (stmt != null) {
                try {
                    stmt.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            if (conn != null) {
                try {
                    conn.close();
                } catch (SQLException e) {
```

```
                    e.printStackTrace();
                }
            }
        }
    }
}
```

工具类调用

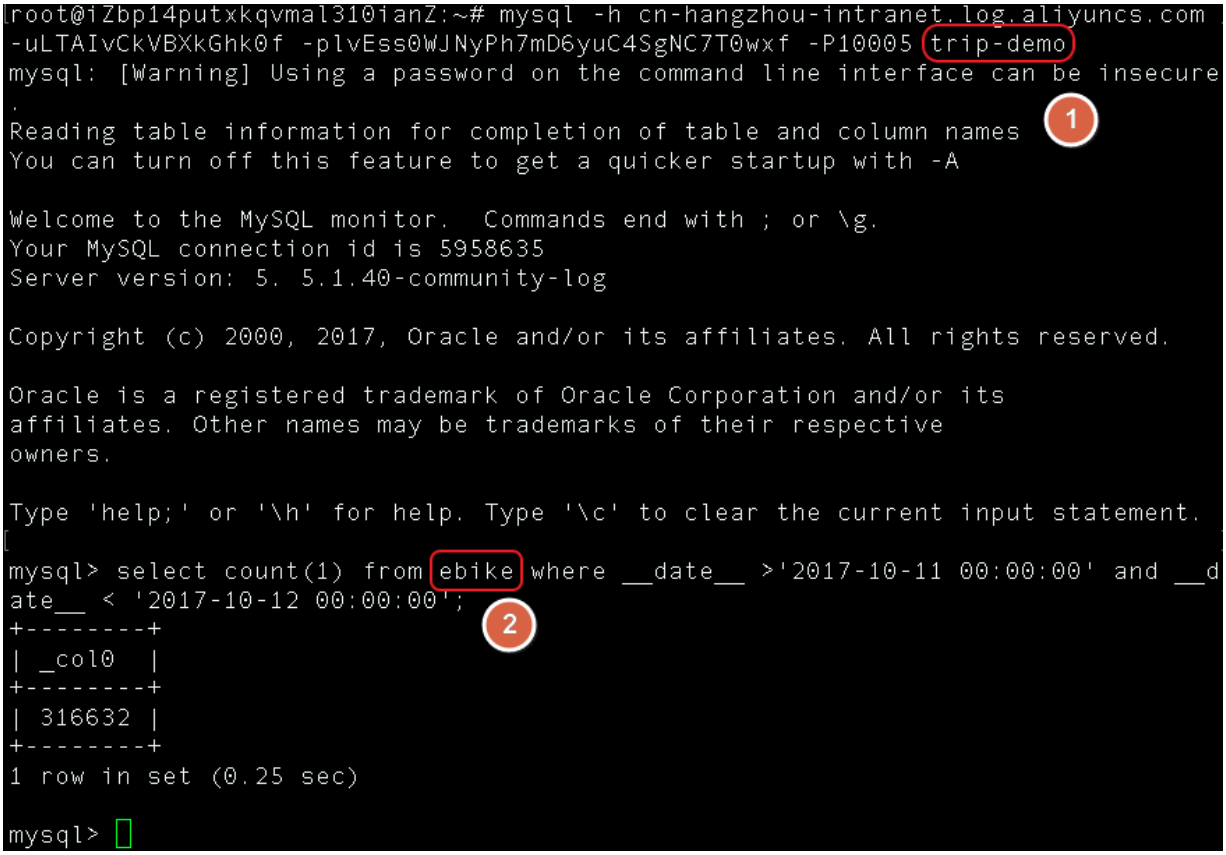
在经典网内网/VPC环境通过MySQL Client进行连接。



说明：

1. ①处填写您的Project。
2. ②处填写您的Logstore。

图 **7-32:** 连接示例



8 可视化分析

8.1 分析图表

8.1.1 图表说明

日志服务提供类似于SQL的聚合计算功能，一切通过SQL聚合计算的结果都可以通过日志服务提供的可视化图表进行渲染。

> 说明：
> 使用可视化图表前，请仔细阅读实时分析简介。

前提条件

1. 创建索引，并开启分析功能。
2. 只有在查询中使用分析语句，才能根据统计结果为您展示图表。

图表类型

目前日志服务提供了如下图表类型：

图 **8-1:** 图表类型



每种图表使用方式请参考如下文档：

- 表格
- 折线图
- 柱状图
- 条形图
- 饼图
- 单值图
- 面积图
- 地图
- 流图
- 桑基图
- 词云

8.1.2 仪表盘

仪表盘是日志服务提供的实时数据分析大盘。您可以将常用的查询语句以图表形式展示，并将多项分析图表保存到仪表盘中。通过仪表盘可以一次性查看多个分析语句的分析图表，当您打开或刷新仪表盘时，这些分析图表会自动执行一遍查询分析语句。

日志服务同时支持控制台分享内嵌功能，您不仅可以在日志服务控制台中查看仪表盘，还可以将某个仪表盘页面外嵌到其他网站页面中，让您的数据分析与数据展示手段更加多样化。另外，添加图表到仪表盘时，还可以设置下钻分析 ，设置之后，在仪表盘页面中单击该图表，可以得到更深维度的分析结果。

限制说明

- 每个Project最多可创建50个仪表盘。
- 每个仪表盘最多可包含50张分析图表。

功能试用

试用链接：请单击此处试用

用户名：sls_reader1@1654218965343050

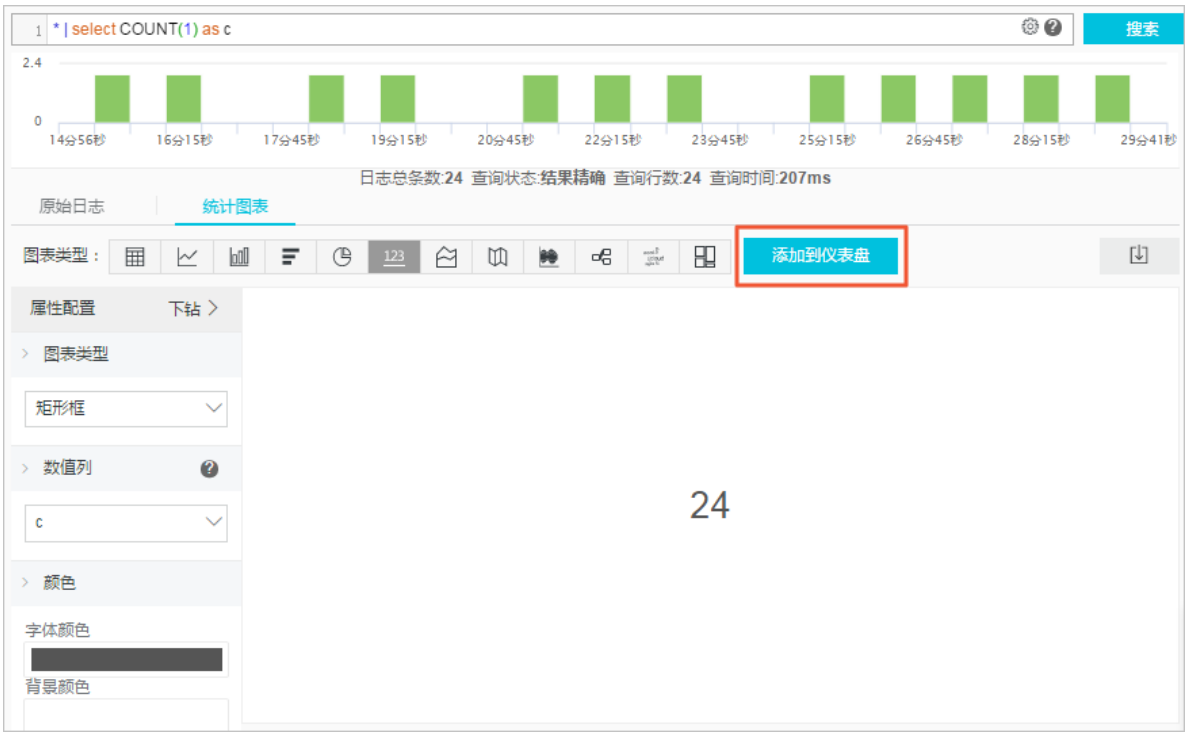
密码：pnX-32m-MHH-xbm

创建仪表盘

1. 登录日志服务控制台，单击Project名称。

2. 在**Logstore**列表页面单击**查询分析**列下的**查询**。
3. 在搜索框中输入查询分析语句，并单击**搜索**。
4. 在**分析图表**页签中添加图表配置。
5. 单击**添加到仪表盘**。
6. 设置仪表盘属性。

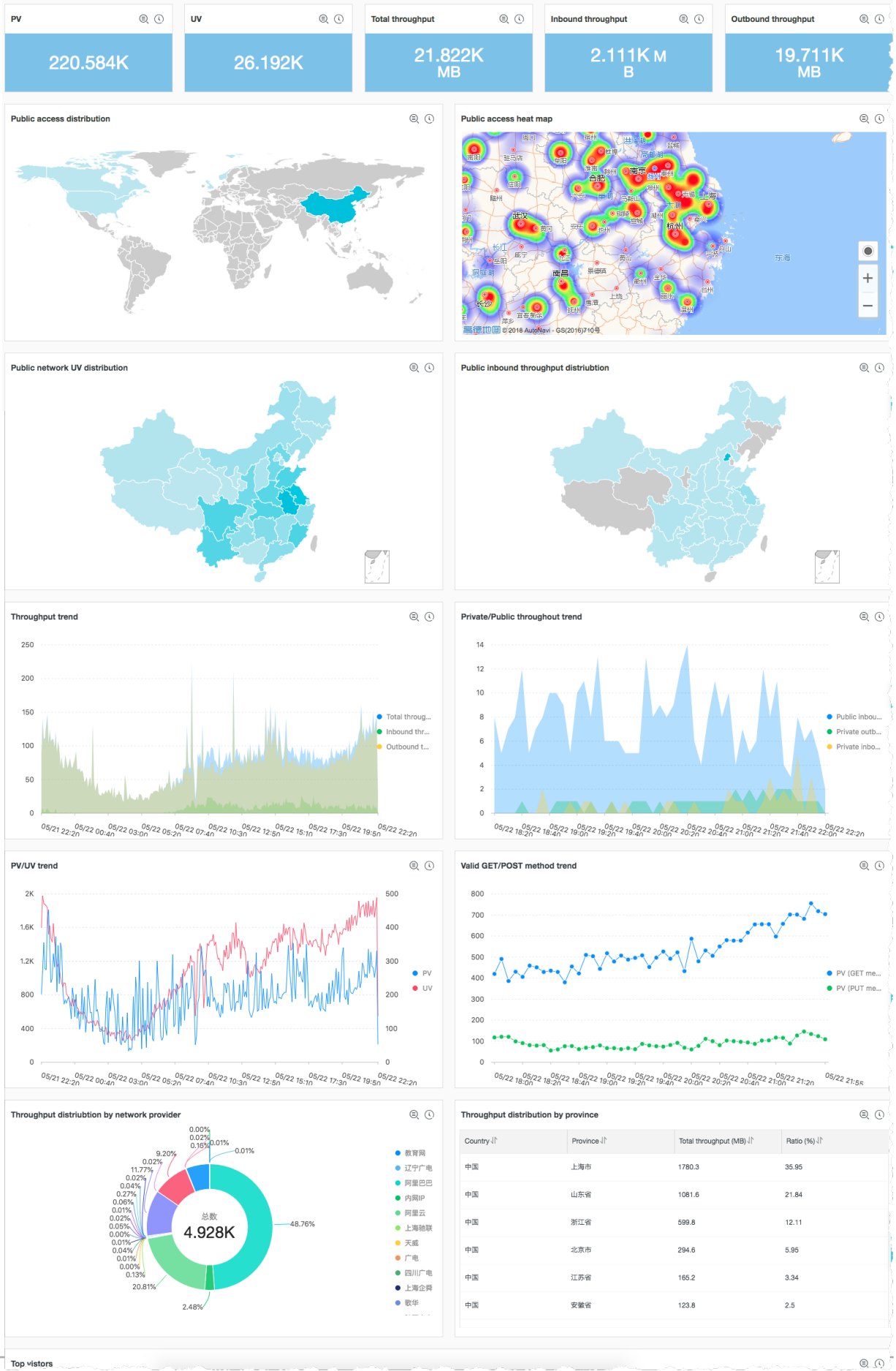
配置	说明
操作类型	• **添加到已有仪表盘**：将当前分析图表添加到已有的仪表盘。 • **新建仪表盘**：将当前分析图表添加到新建的仪表盘中。
仪表盘列表	选择已有的仪表盘名称。 **说明**： 仅在**操作类型**为**添加到已有仪表盘**时指定。
Dashboard名称	新建的仪表盘名称。 **说明**： 仅在**操作类型**为**新建仪表盘**时指定。
图表名称	为当前分析图表命名。该名称会作为图表标题显示在仪表盘页面中。
查询语句	当前分析图表的查询语句。
Token配置	在**查询语句**中选中部分查询语句并单击**生成Token**可生成占位符变量。如果其他图表的下钻事件为跳转到这个仪表盘，单击其他图表时会跳转到这个仪表盘，占位符变量替换为触发下钻事件的图表值，并以替换变量后的查询语句刷新仪表盘。详细信息请查看下钻分析。 • **Token key**：为占位符变量命名。如果占位符变量名称与触发下钻事件的仪表盘图表设置的变量相同时，在下钻事件中占位符变量替换为触发下钻事件的图表值。 • **默认值**：占位符变量在当前仪表盘中的默认值。 • **生成结果**：确认变量配置。



7. 单击**确定**，结束配置。

您可以通过多次添加的方式，将多个分析图表添加到一个仪表盘中。

添加了多个分析图表的仪表盘：


PV
220.584K
UV
26.192K
Total throughput
21.822K MB
Inbound throughput
2.111K MB
Outbound throughput
19.711K MB
Public access distribution
Public access heat map
Public network UV distribution
Public inbound throughput distriubtion
Throughput trend
Total throug...
Inbound thr...
Outbound t...
Private/Public throughout trend
Public inbou...
Private outb...
Private inbo...
PV/UV trend
PV
UV
Valid GET/POST method trend
PV (GET me...
PV (PUT me...
Throughput distriubtion by network provider
总数
4.928K
48.76%
20.81%
2.48%
Throughput distribution by province
Country
Province
Total throughput (MB)
Ratio (%)
中国 上海市 1780.3 35.95
中国 山东省 1081.6 21.84
中国 浙江省 599.8 12.11
中国 北京市 294.6 5.95
中国 江苏省 165.2 3.34
中国 安徽省 123.8 2.5
Top visitors

修改仪表盘

1. 在**Logstore**列表页面单击**查询分析**列下的**查询**。
2. 单击左侧导航栏中的仪表盘名称，直接进入对应仪表盘。
3. 单击仪表盘右上角的**编辑**。

   您可以在仪表盘的编辑视图中进行以下操作：

   - 修改每个分析图表的时间范围、大小和标题。
   - 创建*Markdown*图表。
   - 删除分析图表。
4. 单击右上角**保存**。



查看仪表盘

在查询页面，单击左侧的仪表盘名称可以直接进入对应仪表盘。

- 设置仪表盘的全局时间范围。

  1. 在仪表盘页面单击左上角的**请选择**。
  2. 在弹出选项卡选择时间范围。您可以设置时间范围为**相对时间**、**整点时间**和**自定义时间**。
  3. 单击**确定**。

  说明：

  - 修改时间范围后，所有图表的时间都会改成这个时间范围。
  - 单击页面右上角的**重置时间**可以重置所有图表的时间范围。

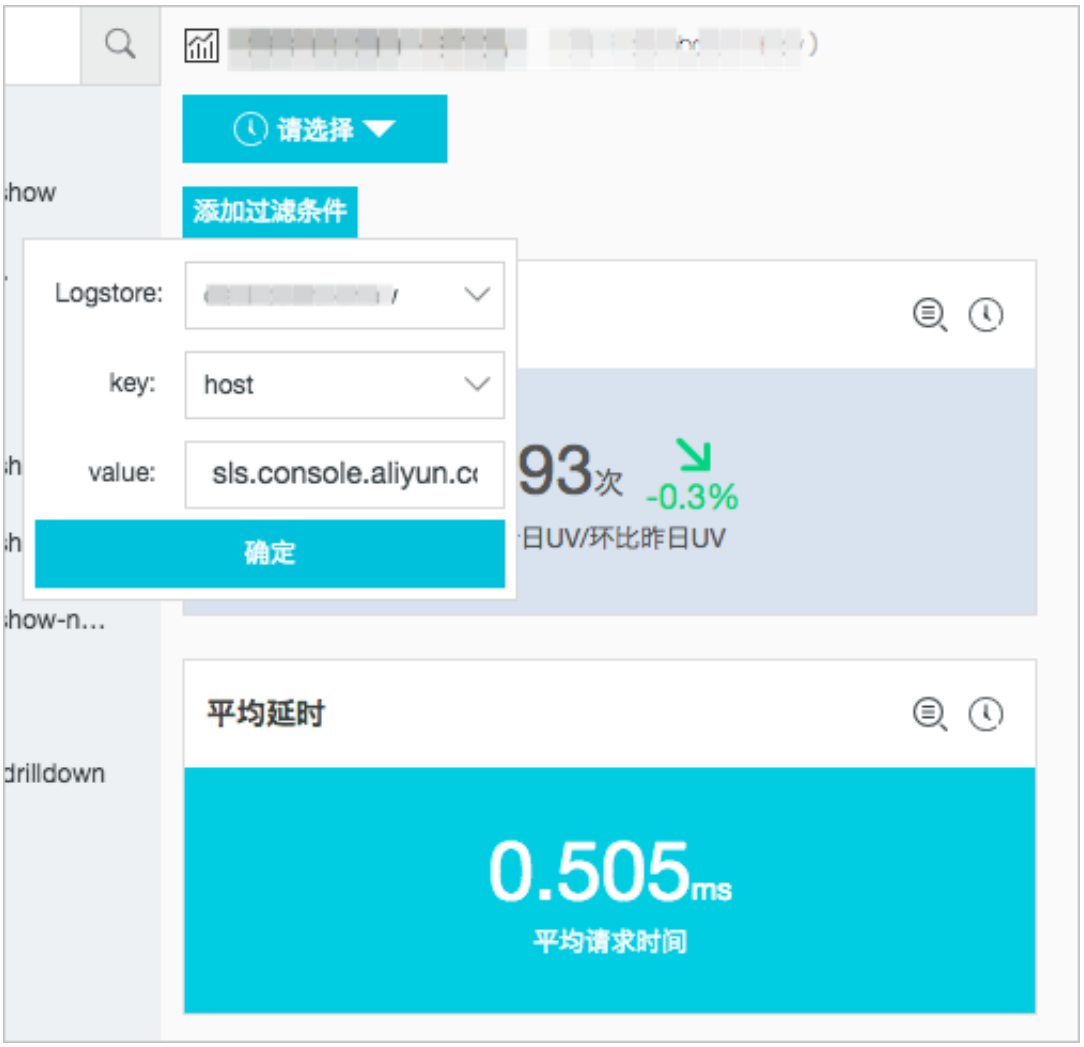
- 时间选择器仅在当前页面提供临时的图表查看方式，系统不保存该设置。您下次查看报表时，系统仍会为您展示默认的时间范围。



- 添加过滤条件，进一步筛选仪表盘显示结果。

  1. 在仪表盘页面单击左上角的添加过滤条件。
  2. 设置**Lostore**、**key**和**value**。

     例如您可以筛选oss这个Logstore中host字段为sls.console.aliyun.com的访问日志信息。



- 设置自动刷新仪表盘。

  1. 在仪表盘页面单击右上角的**自动刷新**。
  2. 设置自动刷新的时间范围。

     仪表盘会根据您设置的时间范围定时自动刷新页面。

删除仪表盘

当您不需要某个仪表盘时，可以删除仪表盘。删除后不可恢复。

1. 在**Logstore**列表页面单击左侧导航栏中的**仪表盘**。
2. 单击对应仪表盘右侧的**删除**。
3. 在弹出提示框中单击**确定**。



8.1.3 仪表盘过滤器

在日志服务仪表盘中增加过滤器配置，可以过滤器缩小查询范围或替换占位符变量，即对整个仪表盘进行查询过滤（Filter）和变量替换（Variables）操作。

日志服务仪表盘中的每一张图表是一个查询分析语句，在仪表盘中增加过滤器也就是为所有图表批量增加过滤条件，或者批量替换所有图表中设置的占位符变量。过滤器配置分为以下两种：

- 过滤器类型：指定key和value，并将其作为过滤条件增加到查询语句`[serch query]`前。新的查询语句为`key: value AND [serch query]`，表示在原查询语句的结果中，查找包含`key:value`的日志。
- 变量替换类型：指定变量占位符，如果仪表盘中有已设置该变量占位符的图表，则将图表查询语句中的该占位符变量替换为选择的value值。

基本构成

每个过滤器图表可以有一个或者多个过滤器构成，每个过滤器主要包含以下元素：

- 过滤器操作Key值。
- Key对应的列表项。

限制说明

- 每个仪表盘最多可以设置5个过滤器。
- 过滤器类型的过滤器中，value可以多选，也可以在请输入中自定义添加一个新的value。多选时，过滤条件之间为`or`关系。

前提条件

1. 已开启并配置索引。
2. 已创建仪表盘，并设置了变量占位符。

操作步骤

1. 登录日志服务控制台，单击Project名称。
2. 在查询分析列中单击查询。
3. 在左侧导航栏中单击指定仪表盘名称。
4. 在仪表盘页面右上角单击添加过滤器。
5. 设置过滤器在仪表盘中的显示配置。

   表 8-1: 过滤器图表配置项

配置项	说明
图表名称	过滤器图表名称。
显示边框	选择显示边框，为过滤器图表增加边框。
显示标题	选择显示标题，会在仪表盘中展示过滤器图表的标题。
显示背景	选择显示背景，为过滤器图表添加白色背景。

6. 单击添加过滤器，设置过滤器，并单击确定。

   表 8-2: 过滤器配置项

配置项	说明
类型	过滤器的类型，包括： • 过滤器 • 变量替换
Key值	• 过滤器类型中，Key值为过滤条件中的key。 • 变量替换类型：Key值为指定的变量占位符。 说明： 变量占位符必须是前提条件中已配置的变量占位符，才能成功替换。
列表项	过滤器中预置的列表项，其中： • 过滤器类型中，列表项表示过滤条件中的value。您可以设置多个value，生成过滤器之后可以在查看仪表盘时根据需求选择value。

配置项	说明
	• 变量替换类型：列表项为指定变量占位符的替换值。您可以设置多个替换值，生成过滤器之后可以在查看仪表盘时根据需求选择替换值。 说明： 在列表项右侧的输入框中输入列表项的值，并单击添加列表项设置列表项。

下拉模式	设置列表项的显示框类型。
	• 开启后，列表项显示为下拉列表格式。
	• 关闭后，列表项显示为选项格式。
	  • 过滤器类型为多选（CheckBox）。
	  • 变量替换类型为单选（Radio）。



当前仪表盘页面会自动刷新显示新的过滤器配置。根据您的需求在请选择中选择value或占位符的替换值，并单击添加。

过滤器类型的过滤器中，value可以多选，也可以在请输入中自定义添加一个新的value。多选时，过滤条件之间为or关系。

应用场景

过滤器多用于在当前仪表盘中动态修改查询条件和对图表中已经存在的变量占位符进行变量替换。每一张图表实际为一个查询分析语句，满足 `[search query] | [sql query]` 的形式，过滤器实质上会操作该查询分析语句。

- 如果为过滤器，则会在`[serch query]`前加上过滤的值，以AND连接为新的查询语句，即`key: value AND [serch query]`
- 如果变量替换过滤器，则会查询整个仪表盘存在变量占位符的图表，将对应名称的变量占位符替换为选择的`value`值

示例

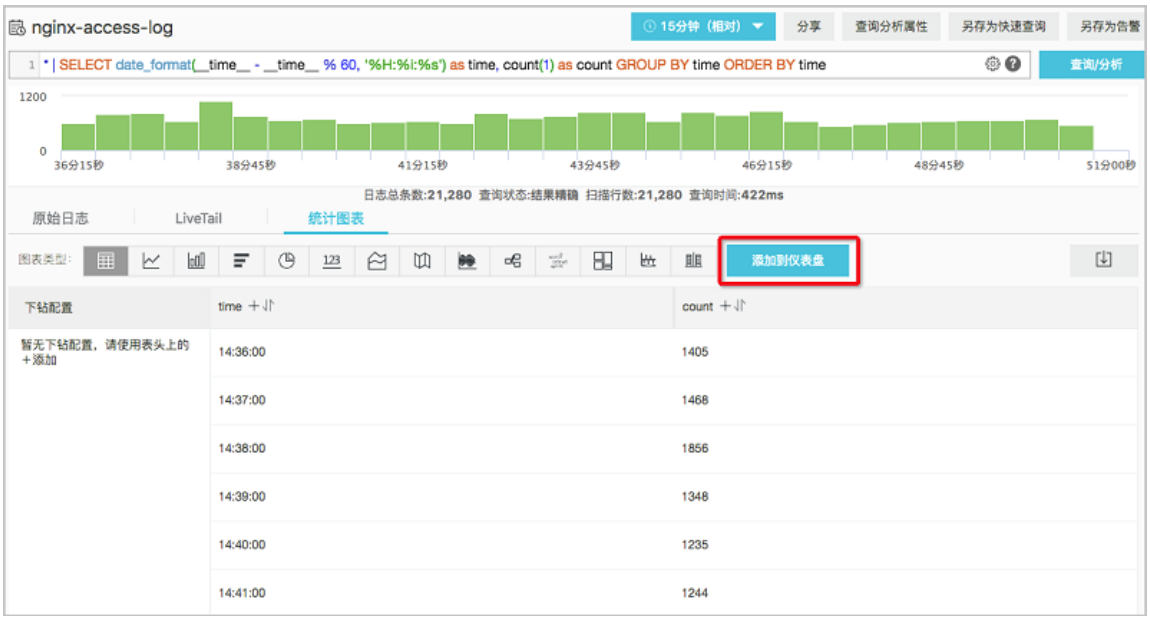
例如，采集*Nginx*日志后，需要对采集到的日志数据进行实时查询与分析。

- **场景1**：基于不同时间粒度

  通过分析语句可以查看每分钟的访问PV，当需要查看秒级别的数据时，需要调整`__time__ - __time__ % 60`的值，传统做法为修改查询分析语句，多次查询时操作繁琐。此时可以通过过滤器完成变量替换。

  1. 通过以下语句查看分钟访问PV的数据。

     ```
     * | SELECT date_format(__time__ - __time__ % 60, '%H:%i:%s') as time, count(1) as count GROUP BY time ORDER BY time
     ```

  2. 将分析图表添加到仪表盘，并选中60作为变量占位符的默认值，变量名为`interval`。



3. 添加过滤器，并设置类型为变量替换。其中：

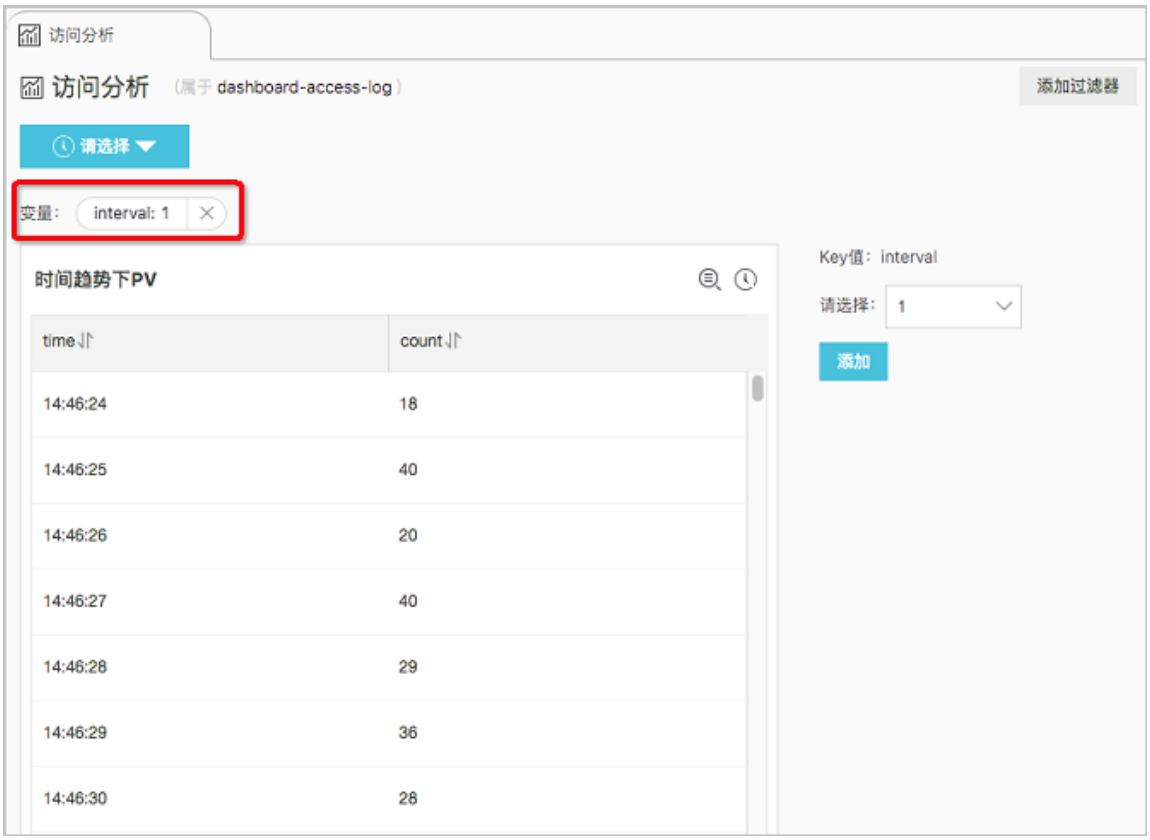
   - 类型为变量替换。
   - **Key**值为`interval`。
   - 列表项为`1`（表示每秒）和`120`（表示每2分钟）。



4. 在过滤器中选择1，此时仪表盘为秒级别的粒度。

   替换变量后的查询语句：

```
* | SELECT date_format(__time__ - __time__ % 1, '%H:%i:%s') as
time, count(1) as count GROUP BY time ORDER BY time
```



- **场景2**：动态切换过滤方法

  通过过滤器还可以动态切换不同的请求方法（request_method）。场景1中，查询语句为*，表示不设置任何过滤条件，即所有的日志都在查询范围之中。此时，可以再添加一个过滤器便于查看不同request_method的访问情况。

  1. 在场景1中的仪表盘中增加过滤器，设置如下。

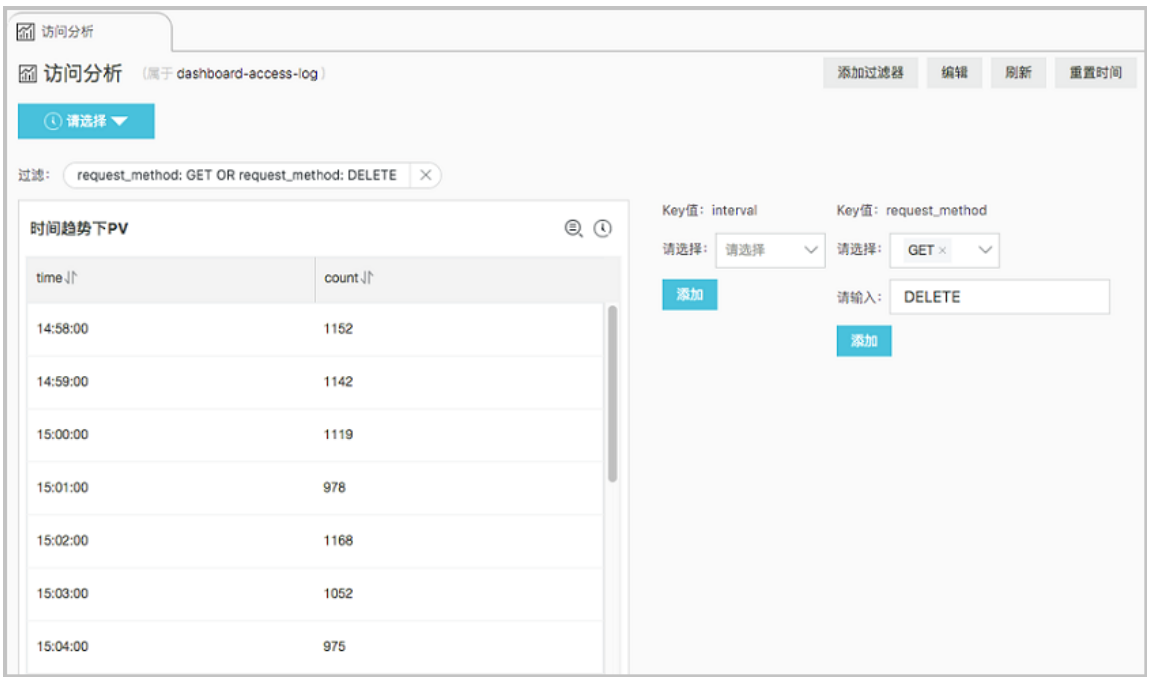
     - 类型为过滤器。
     - **Key**值为request_method。
     - 列表项为列表项：GET、POST和PUT。



2. 在过滤器的下拉列表项中选择`GET`，并手动输入`DELETE`。

   图表中只显示`request_method`为`GET`和`DELETE`的访问。实质上查询分析语句已经变为：

```
(*) and (request_method: GET OR request_method: DELETE) | SELECT
date_format(__time__ - __time__ % 60, '%H:%i:%s') as time, count(1
) as count GROUP BY time ORDER BY time
```



8.1.4 下钻分析

日志服务分析图表除了提供最基本的数据可视化能力之外，还提供了向下钻取（drill down）的功能，您可以在添加一个图表到仪表盘的时候，通过改变下钻列表中的各个配置项，从而使得仪表盘中的分析图表具备更强大的功能。

钻取是在数据分析中不可缺少的功能之一，通过改变展现数据维度的层次、变换分析的粒度从而关注数据中更详尽的信息。它包括向上钻取（roll up）和向下钻取（drill down）。上钻是沿着维度的层次向上聚集汇总数据，下钻是在分析时加深维度，对数据进行层层深入的查看。通过逐层下钻，数据更加一目了然，更能充分挖掘数据背后的价值，及时做出更加正确的决策。

日志服务支持对仪表盘中分析图表的下钻分析，设置下钻的维度和层次后，可以在仪表盘中通过鼠标点击数据点跳转到更深维度的分析页面。仪表盘中的分析图表实际上是查询语句的结果，如果为请求状态表格设置下钻分析、并添加到仪表盘，在仪表盘中单击某个请求状态类型，可以查看请求状态为特定类型的日志信息。

限制说明

日志服务中，支持下钻分析的图表包括：

- 表格
- 线图
- 柱状图
- 条形图
- 饼图
- 单值图
- 面积图
- 矩形树图

前提条件

1. 已开启并配置索引。
2. 已配置要跳转到的快速查询、仪表盘和自定义链接。
3. 在跳转到的快速查询和仪表盘配置中配置查询语句变量占位符。详情请参考快速查询和仪表盘。

配置步骤

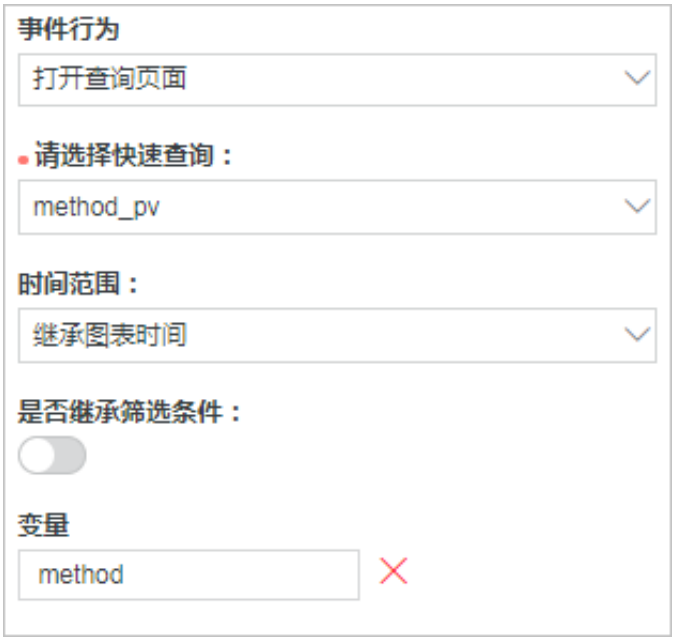
1. 登录日志服务控制台，单击Project名称。
2. 在**Logstore**列表的**查询分析**列下单击**查询**。

3. 输入您的查询分析语句，设置时间范围，并单击**搜索**。
4. 在**统计图表**页签中选择**图表类型**，并设置图表**属性**。
5. 单击**属性配置**栏右侧的**下钻**，设置下钻事件行为。

   下钻配置默认为关闭状态，下钻事件通过单击触发。下钻事件行为指在仪表盘页面中单击分析图表而触发的事件。设置下钻事件后，在仪表盘中单击这张图表中的数据，根据您配置的事件行为，自动跳转到对应页面。您可以选择以下4种配置。

   - **不开启**：表示不开启下钻功能。
   - **打开查询页面**：表示开启下钻功能，下钻事件为打开查询页面。

     单击图表内容时，会用单击的图表值替换快速查询语句中设置的占位符，基于图表值进行更深层次的查询。

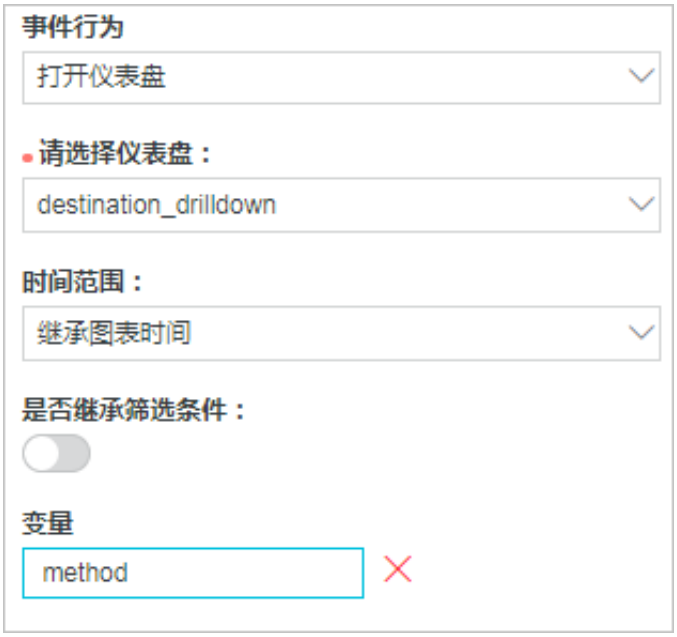


配置	说明
快速查询	需要跳转到的快速查询名称。如何配置快速查询请参考快速查询。
时间范围	设置跳转到的快速查询的时间范围。默认为**继承图表时间**，即仪表盘页面中单击图表跳转到快速查询后，查询语句对应的时间范围默认为触发事件时仪表盘中设置的图表的时间。
是否继承筛选条件	如果选择**继承筛选条件**，则会把触发事件仪表盘中添加的筛选条件同步到快速查询中，并以AND的方式添加到查询语句之前。

配置	说明
变量	单击添加变量即可输入变量占位符名称。当快速查询中变量和本次添加的变量名称一致时，会将查询语句中的变量替换为触发下钻事件的图表值，从而灵活改变快速查询的维度。 说明： 如果选择添加变量，则需要事先在跳转到的快速查询中配置查询语句变量占位符。

- **打开仪表盘**：表示开启下钻功能，下钻事件为打开仪表盘。

  仪表盘中的图表实际上是查询语句的图表形式的结果，预先在跳转到的仪表盘图表查询语句中设置占位符，单击上层仪表盘中的图表内容时，会用单击的图表值替换预设的占位符，基于图表值进行更深层次的查询。



配置	说明
仪表盘	需要跳转到的目标仪表盘名称，如何配置仪表盘请参考仪表盘。
时间范围	设置跳转到的仪表盘全局时间范围。默认为**继承图表时间**，即仪表盘页面中单击图表跳转到仪表盘后，仪表盘的时间范围默认为触发事件的仪表盘图表设置的时间范围。
是否继承筛选条件	如果选择**继承筛选条件**，则会把触发事件仪表盘中添加的筛选条件同步到跳转到的仪表盘中，并以AND的方式添加到查询语句之前。

配置	说明
变量	单击添加变量即可输入变量占位符名称。当跳转到的仪表盘中的分析图表查询语句变量和本次添加的变量名称一致时，会将分析图表查询语句中的变量替换为触发下钻事件的图表值，从而灵活改变目标仪表盘中分析图表的查询语句。 说明： 如果选择添加变量，则需要事先在跳转到的仪表盘中配置查询语句变量占位符。

- 自定义**http**链接：表示开启下钻功能，下钻事件为打开自定义http链接。

  http链接中的路径部分表示访问的目的端文件的层级路径，您可以在定义http链接的路径部分添加可选参数域，单击仪表盘中的图表内容时，会用图表值替换http链接中的参数，跳转到重新定位的http链接中。



配置	说明
链接地址	需要跳转到的目标地址。
可选参数域	单击可选参数变量，可以将链接地址中的某一部分替换为触发下钻事件的图表值。

6. 单击**添加到仪表盘**，配置仪表盘，并单击**确定**。

   后续您可以在**仪表盘**页面中查看该分析图表，单击图表即可查看更深层次的分析结果。

示例

例如，在名为accesslog的Logstore中存放采集到的Nginx访问日志，名为RequestMethod的仪表盘中展示Nginx日志的常见分析场景，名为destination_drilldown的仪表盘展示PV随时间分布的趋势。您可以为请求方法的分类表格设置下钻分析，并将其添加到RequestMethod仪表盘中，并将下钻事

件设置为跳转到destination_drilldown仪表盘。在RequestMethod仪表盘中单击各个请求方法即可跳转到destination_drilldown仪表盘查看对应的PV趋势。

流程如下：

1. 设置跳转到的仪表盘（**destination_drilldown**）。
   1. 根据请求类型筛选日志，并查看PV随时间的变化。

      查询语句：

      ```
      request_method: * | SELECT date_format(date_trunc('minute', __time__), '%H:%i:%s') AS time, COUNT(1) AS PV GROUP BY time ORDER BY time
      ```

   2. 通过折线图表示查询结果，并将折线图保存到仪表盘中。

      保存到仪表盘时，将*设置为占位符，并命名为method，如果跳转到这个快速查询的下钻事件变量同样为method，即可用单击的图表值替换*，再次执行查询分析。

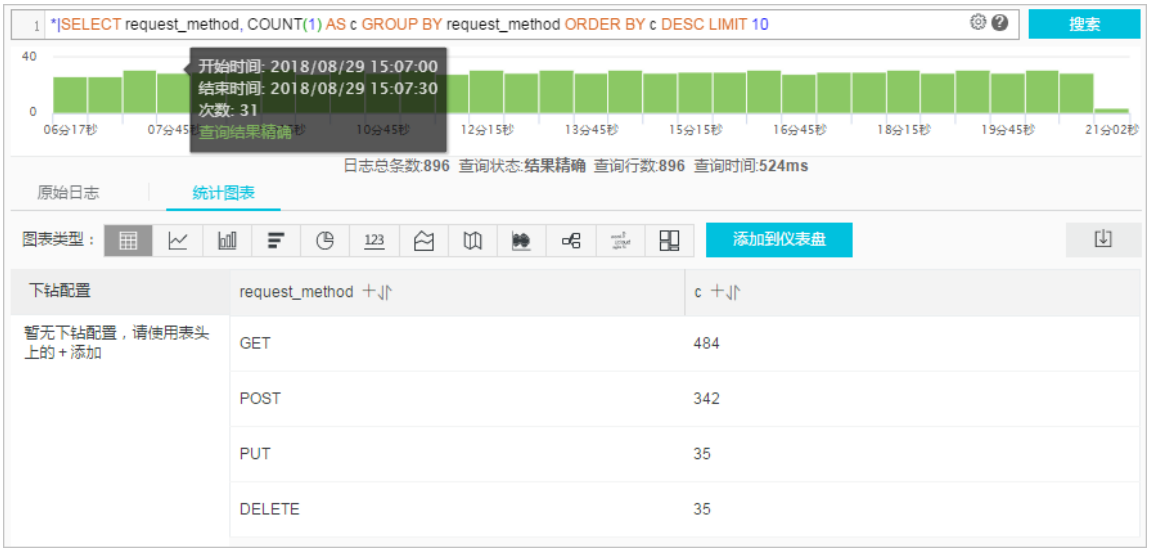
      

2. 设置触发下钻分析的图表，并将其添加到仪表盘（**RequestMethod**）。

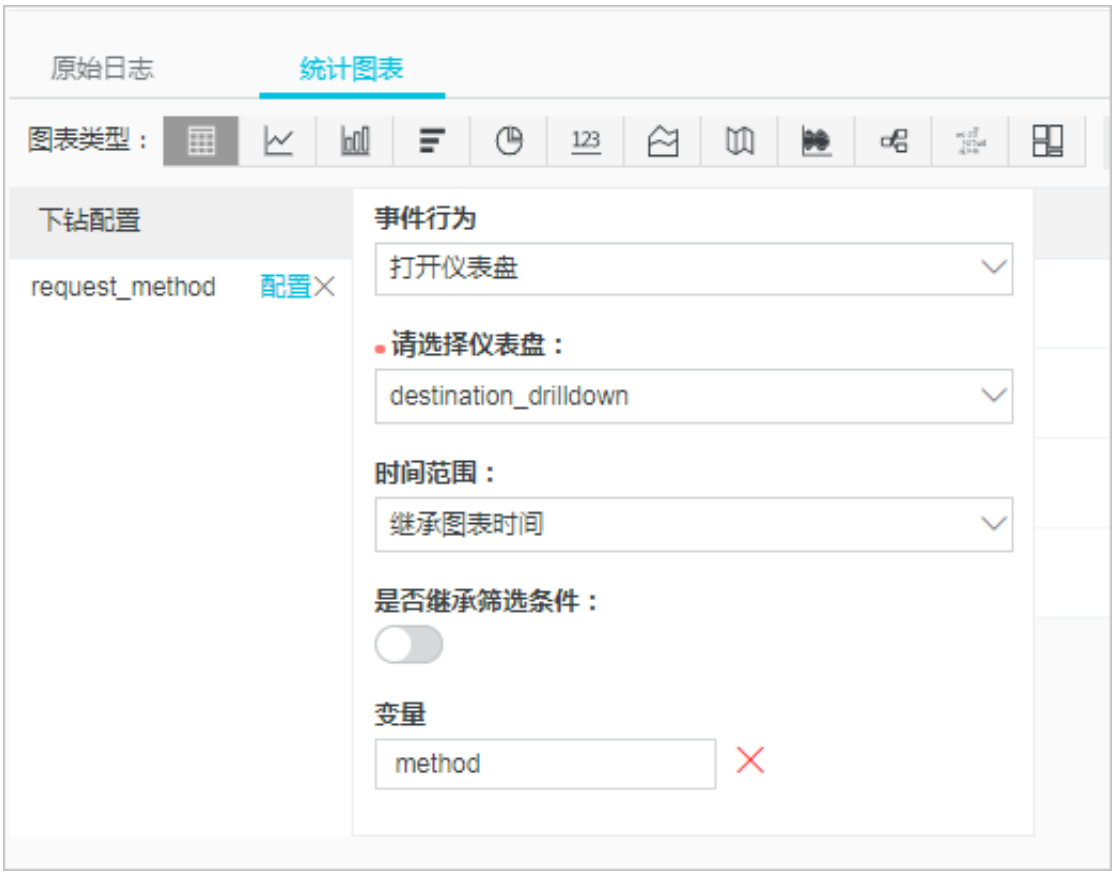
1. 在查询页面通过SQL语句分析Nginx访问日志中各种请求方法（request_method）的日志条数，并将结果以表格形式表示。

```
*|SELECT request_method, COUNT(1) AS c GROUP BY request_method
ORDER BY c DESC LIMIT 10
```

查询结果：



2. 为request_method一列设置下钻分析：

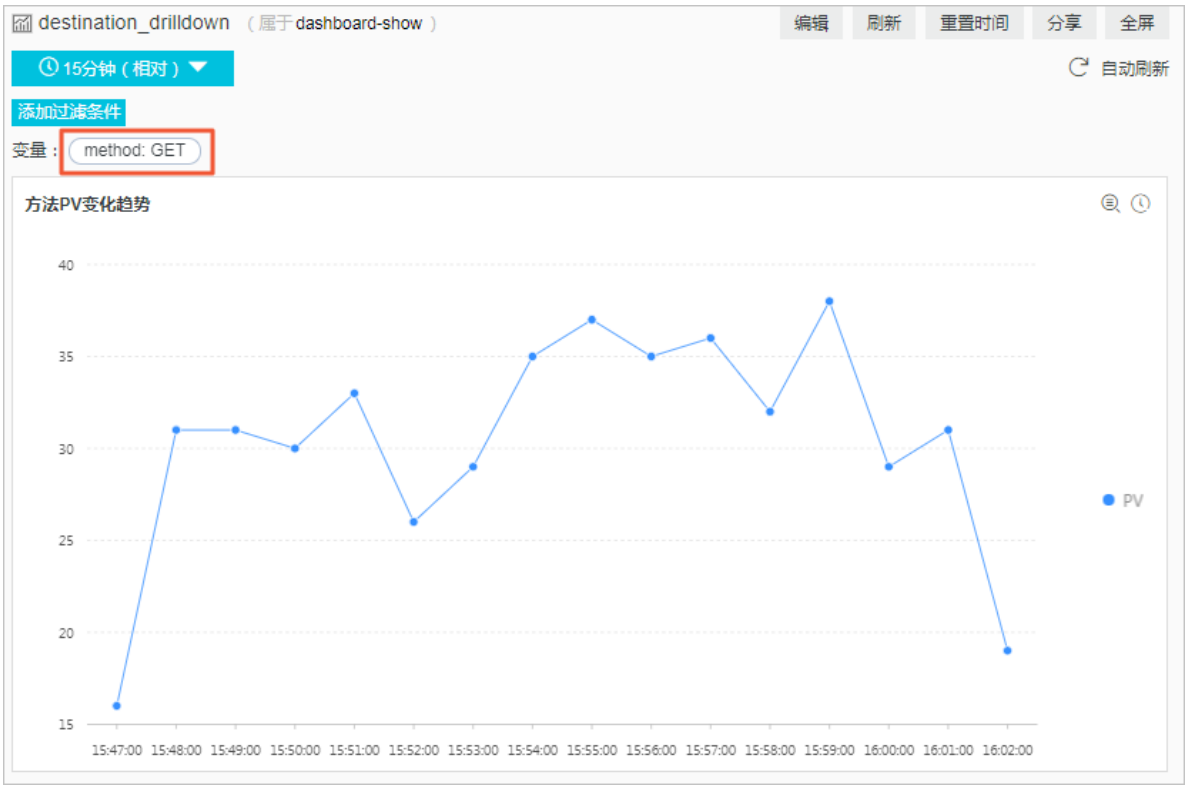


3. 在**RequestMethod**仪表盘中单击**GET**请求。



4. 成功跳转到**destination_drilldown**仪表盘。

   页面自动跳转到*1*中设置的仪表盘，原查询语句中的`*`已替换为单击的图表值`GET`，表示查看GET请求PV随时间的变化。



8.1.5 Markdown图表

日志服务支持在仪表盘中增加Markdown图表，在Markdown图表插入图片、链接、视频等多种元素，使您的仪表盘页面更加友好。

在查询分析日志数据时，将多个分析图表添加到仪表盘中，有利于您快速查看多项分析结果、实时监控多项业务的状态信息。日志服务还支持在仪表盘中增加Markdown图表，该图表使用Markdown语言编辑。您可以在Markdown中插入图片、链接、视频等多种元素，使您的仪表盘页面更加友好。

Markdown图表都是根据不同的需求来创建的。您可以在Markdown图表中插入背景信息、图表说明、页面注释和扩展信息等文字内容，优化仪表盘的信息表达；插入快速查询或其他Project的仪表盘链接，方便其他查询页面的跳转；插入自定义的图片，让您的仪表盘信息更加丰富、功能更为灵活。

应用场景

通过Markdown图表可以自定义跳转链接到当前Project的其他仪表盘，同时还可以插入图片以便快速区分。当需要对图表参数进行介绍时，也可以插入Markdown图表进行说明。

图 8-2: 应用场景



前提条件

1. 已成功采集到日志数据。
2. 已配置仪表盘。

操作步骤

1. 在仪表盘页面单击右上角的编辑按钮。
2. 在编辑状态下，单击创建**markdown**图表。
3. 在弹出页面中设置Markdown图表属性。

配置项	说明
图表名称	您创建的Markdown图表名称。
显示边框	选择显示边框，为您的Markdown图表增加边框。
显示标题	选择显示标题，会在仪表盘中为您展示Markdown图表的标题。
显示背景	选择显示背景，为您的Markdown图表添加白色背景。

4. 编辑**Markdown**内容。

在**markdown**内容中输入您的Markdown语句，右侧的图表展示区域会实时为您展示预览界面。您可以根据预览内容调整Markdown语句。

5. 配置完成后单击确定。

图 **8-3:** 创建**Markdown**图表



配置完成后，您刚刚创建的Markdown图表会出现在当前仪表盘的下方位置。

修改Markdown图表

- 修改图表位置和大小

  1. 在仪表盘页面单击右上角的编辑。
  2. 鼠标拖动Markdown图表到指定位置，拖动图表右下角调整图表大小。
  3. 单击页面右上角的保存。

- 修改图表标题

  1. 在仪表盘页面单击右上角的编辑。
  2. 在图表标题框中输入新的标题。
  3. 单击仪表盘页面右上角的保存，在弹出对话框中单击确定。

- 修改图表内容

  1. 在仪表盘页面单击右上角的编辑。
  2. 单击Markdown图表右上角的编辑。
  3. 修改图表配置，并单击确定。

- 删除图表

  1. 在仪表盘页面单击右上角的编辑。
  2. 单击Markdown图表右上角的删除。
  3. 单击仪表盘页面右上角的保存，在弹出对话框中单击确定。

常用Markdown语法

- 标题

  Markdown语句：

  ```
  # 一级标题
  ## 二级标题
  ```

```
### 三级标题
```

图 **8-4:** 标题预览



- 链接

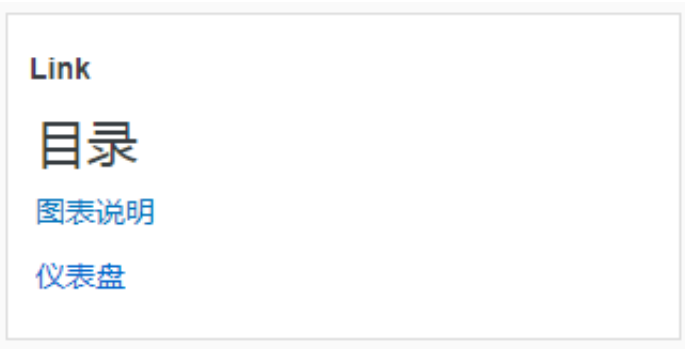
  Markdown语句：

```
### 目录

[图表说明](https://help.aliyun.com/document_detail/69313.html)

[仪表盘](https://help.aliyun.com/document_detail/59324.html)
```

图 **8-5:** 链接预览



- 图片

  Markdown语句：

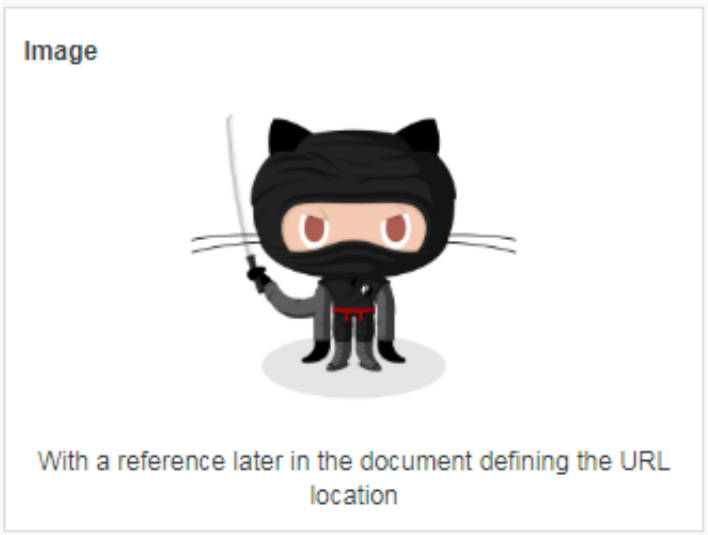
```
<div align=center>

![Alt txt][id]
```

```
With a reference later in the document defining the URL location

[id]: https://octodex.github.com/images/dojocat.jpg  "The Dojocat"
```

图 **8-6:** 图片预览



- 特殊标记

  Markdown语句：

```
---

__Advertisement :)__

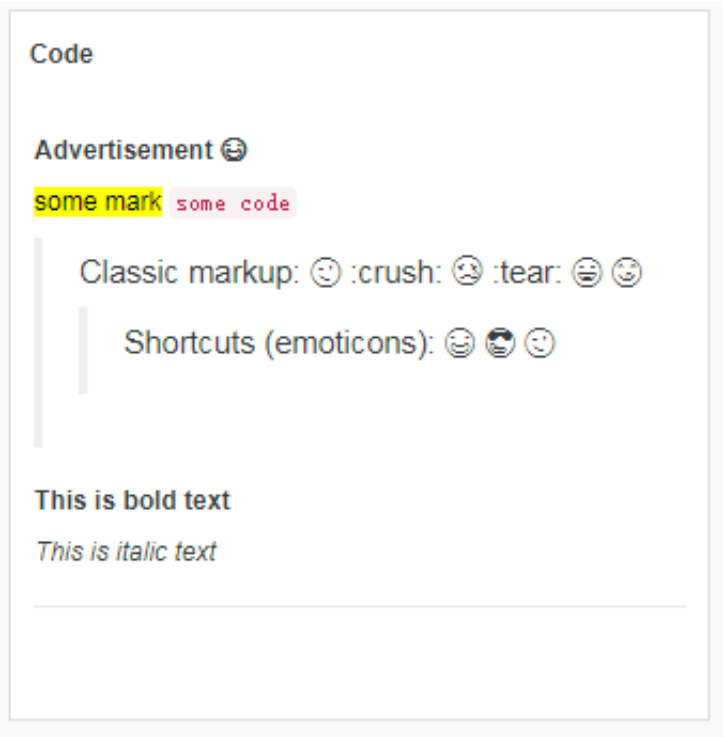
==some mark==    `some code`
> Classic markup: :wink: :crush: :cry: :tear: :laughing: :yum:
>> Shortcuts (emoticons): :-)  8-) ;)

__This is bold text__

*This is italic text*
```

```
---
```

图 8-7: 特殊标记预览



关于Markdown语法的详细说明，请查看*Markdown*语法。

8.1.6 表格

表格作为最常见的数据展示类型，是组织整理数据最基本的手段，表格通过的对数据的组织整理，以便达到快速引用和分析的目的。日志服务提供类似于SQL的聚合计算功能，通过查询分析语法得到的数据结果默认以表格方式进行展示。

基本构成

- 表头
- 行
- 列

其中：

- `SELECT`项的个数为列数。
- 行数由当前时间区间日志条数经过计算后的个数决定，默认为`LIMIT 100`。

使用步骤

1. 在查询页面的查询框中输入查询分析语句，选择时间区间后点击右侧的**查询**按钮。
2. 打开统计图表页签，默认以表格  形式展示结果。

示例

筛选原始日志数据中的列，例如原始日志如下：

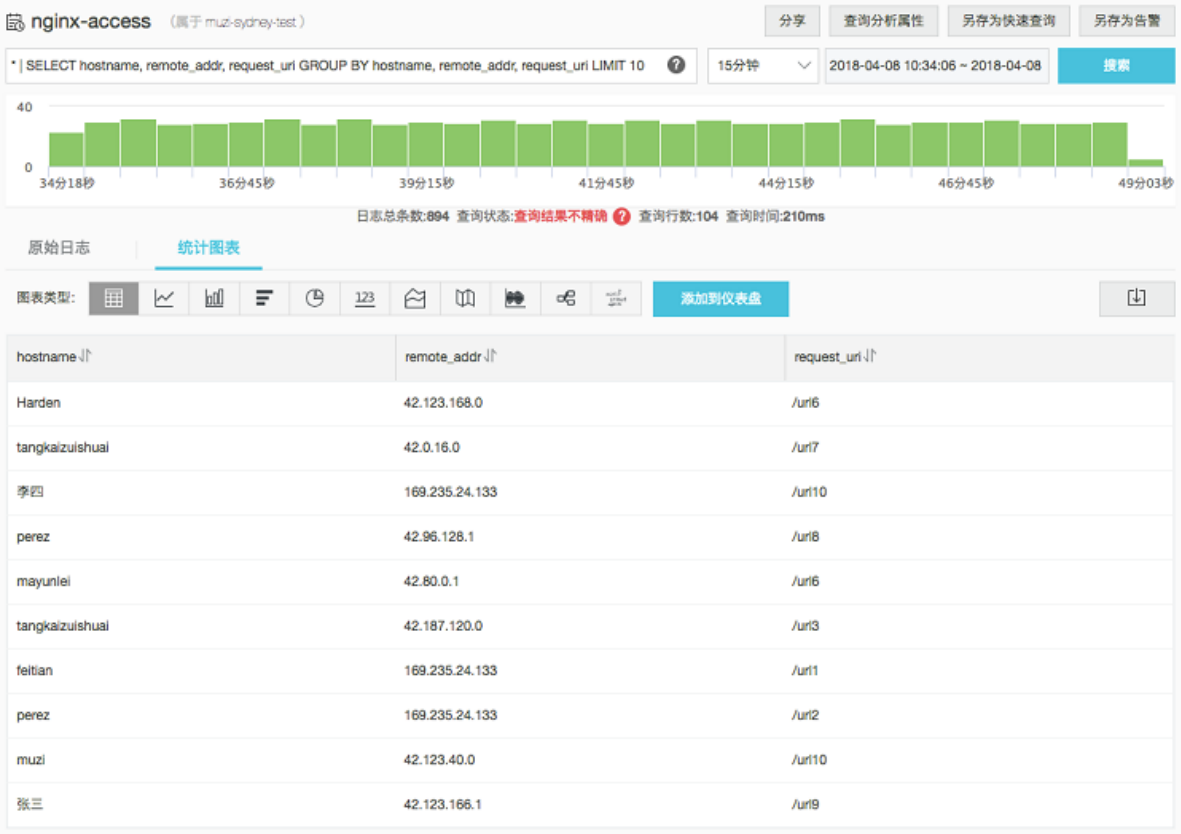
图 8-8: 原始日志



1. 筛选其中最近10条日志的`hostname`、`remote_addr`和`request_uri`。

```
* | SELECT hostname, remote_addr, request_uri GROUP BY hostname,
remote_addr, request_uri LIMIT 10
```

图 **8-9: case 1**



2. 计算单个数据，如当前时间区间`request_time`平均值（平均请求时间），并保留3位小数。

```
* | SELECT round(avg(request_time), 3) as average_request
```

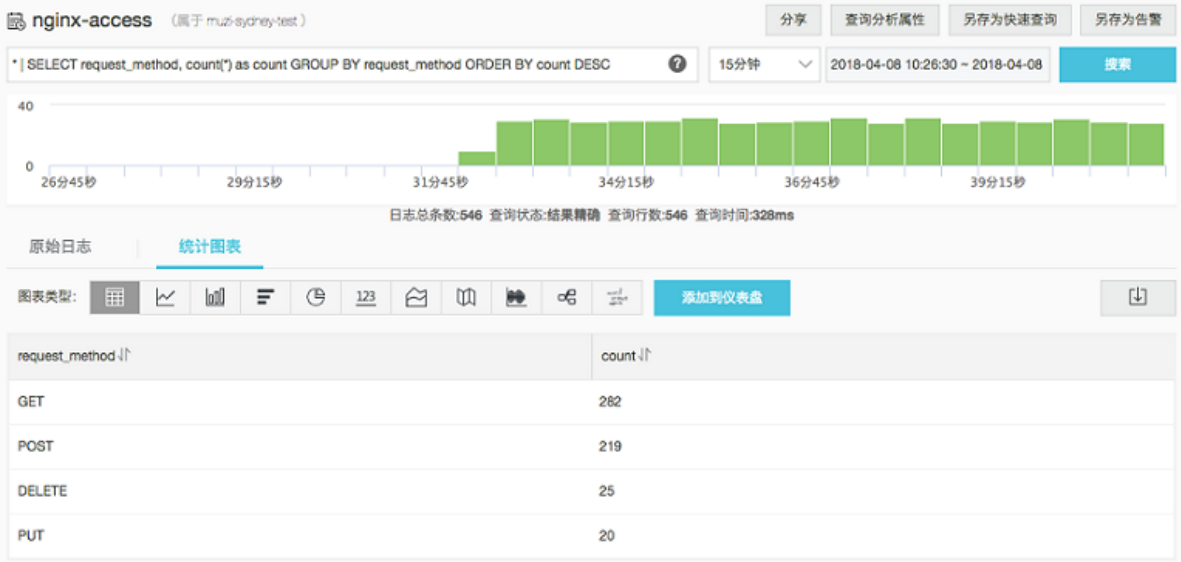
图 **8-10: case 2**



3. 计算分组数据，如当前时间区间request_method分布情况，并降序排列。

```
* | SELECT request_method, count(*) as count GROUP BY request_method
 ORDER BY count DESC
```

图 8-11: case 3



8.1.7 折线图

线图属于趋势类分析图表，一般用于表示一组数据在一个有序数据类别（多为连续时间间隔）上的变化情况，用于直观分析数据变化趋势。

在线图中，我们可以清晰的观测到数据在某一个周期内的变化，主要反映在：

- 递增性或递减性
- 增减的速率情况
- 增减的规律（如周期变化）
- 峰值和峰谷

所以，线图是用于分析数据随时间变化趋势的不二选择。同时，也可以绘制多条线用于分析多组数据在同一时间周期的变化趋势，进而分析诸如数据之间的相互作用和影响（如同增同减，成反比等）。

基本构成

- X轴
- 左Y轴

- 右Y轴（可选）
- 数据点
- 变化趋势线
- 图例

配置项

配置项	说明
X轴	一般为有序数据类别（时间序列）。
左Y轴	可以配置一列或多列数据对应到左轴数值区间。
右Y轴	可以配置一列或多列数据对应到右轴数值区间（右轴图层高于左轴）。
为柱列	将已选择的左Y轴或者右Y轴中的一列以柱状形式表示。
图例位置	图例在图表中的位置，可以配置为上、下、左和右。
间距	坐标轴距离图表边界距离。

使用步骤

1. 键入查询语句，选择时间区间后点击右侧**查询**按钮。
2. 选择折线图  。
3. 配置图表属性。

> **说明：**
> 线图单条线的数据记录数要大于2，以免无法分析数据趋势，同时，建议同一个图上不要超过5条线。

示例

简单折线图

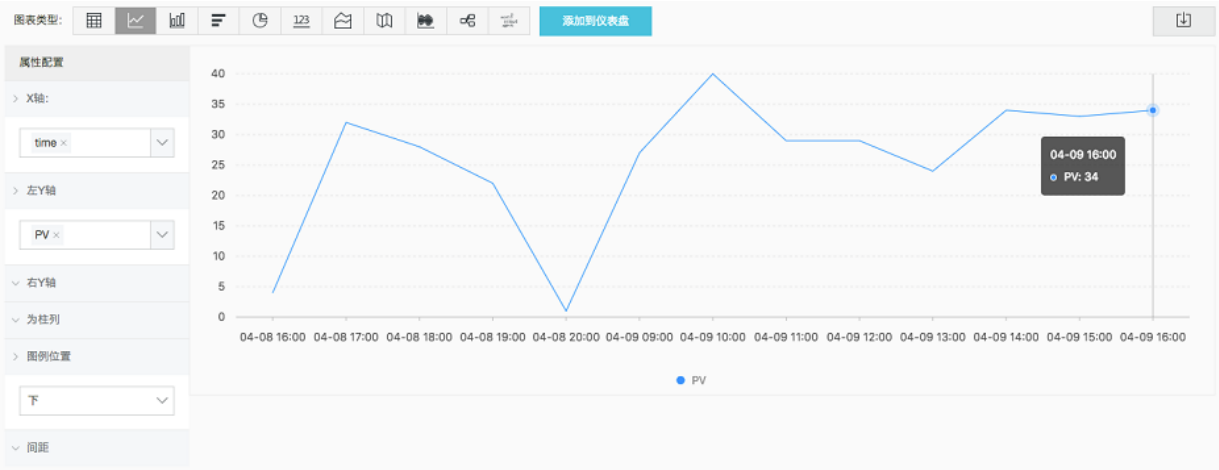
查询`42.0.192.0`这个IP在最近1天内的访问情况：

```
remote_addr: 42.0.192.0 | select date_format(date_trunc('hour',
__time__), '%m-%d %H:%i')
```

```
as time, count(1) as PV group by time order by time limit 1000
```

X轴选择time，左Y轴选择PV并调整图例位置为下方显示，合理改变间距。

图 **8-12:** 简单折线图



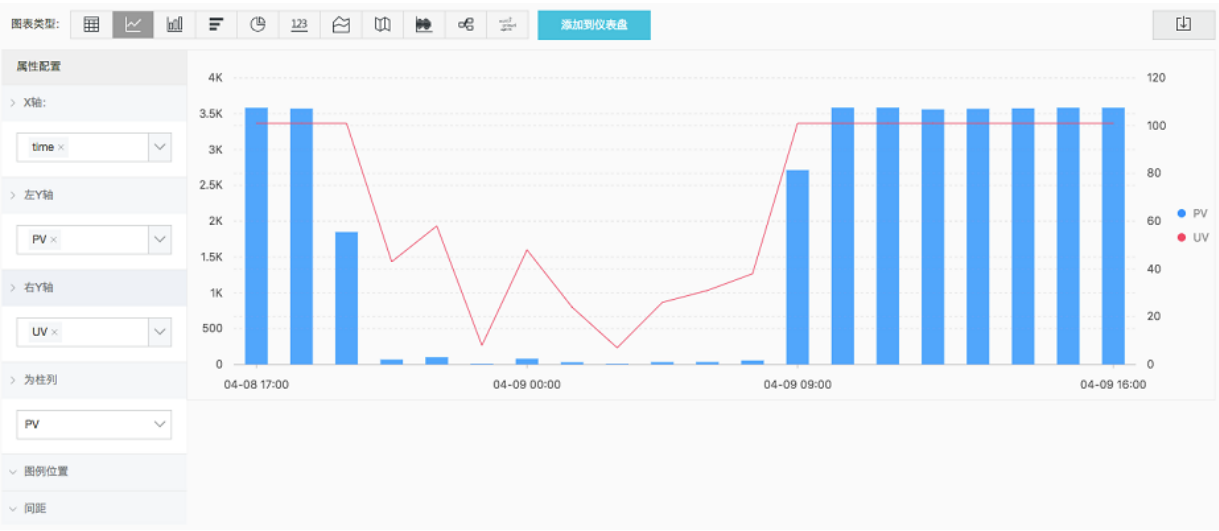
双轴折线图

查询最近1天内的访问PV、UV：

```
* | select date_format(date_trunc('hour', __time__), '%m-%d %H:%i')
as time, count(1) as PV, approx_distinct(remote_addr) as UV group by
time order by time limit 1000
```

X轴选择time，左Y轴选择PV，右Y轴选择UV并指定PV为柱状显示。

图 **8-13:** 双轴折线图



8.1.8 柱状图

柱状图使用垂直或水平的柱子显示类别之间的数值比较，和折线图的不同之处在于，柱状图描述分类数据，并统计每一个分类中的数量，而折线图描述有序数据。

同时，您也可以绘制多个矩形对应同一个分类属性，分为分组和层叠两种模式，进而分析该分类数据在不同维度上的区别。

基本构成

- X轴（横轴）
- Y轴（纵轴）
- 矩形块
- 图例

日志服务提供的柱状图，默认采用垂直柱子，即矩形块宽度一定，高度代表数值大小。有多列数据映射到Y轴时，采用分组柱状形式显示。

配置项

配置项	说明
X轴	一般为分类数据。
Y轴	可以配置一列或多列数据对应到左轴数值区间。
图例位置	图例在图表中的位置，可以配置为上、下、左和右。
间距	坐标轴距离图表边界距离

使用步骤

1. 键入查询语句，选择时间区间后点击右侧**查询**按钮。
2. 选择柱状图 
3. 配置图表属性。

> **说明：**
> 柱状图适用于不超过20条的数据，建议使用LIMIT进行控制，以免横向宽度过宽导致分析对比情况不直观。同时，当有多列数据映射到Y轴时，建议不要超过5个。

示例

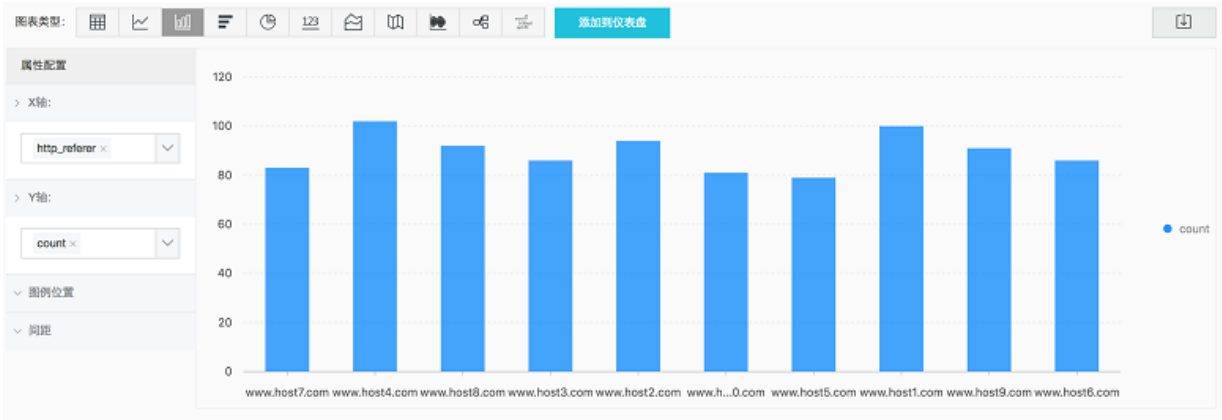
简单柱状图

查看当前时间区间每种`http_referer`的访问次数。

```
* | select  http_referer, count(1) as count group by http_referer
```

X轴选择`http_referer`，Y轴选择`count`。

图 **8-14:** 简单柱状图



分组柱状图

查看当前时间区间每种`http_referer`的访问次数和平均字节数。

```
* | select  http_referer, count(1) as count, avg(body_bytes_sent) as
avg group by http_referer
```

X轴选择`http_referer`，Y轴选择`count`和`avg`。

图 **8-15:** 分组柱状图



8.1.9 条形图

条形图是柱状图另一种形式，即横向柱状图。条形图通常用于分析Top场景，配置方式也和柱状图类似。

基本构成

- X轴（纵轴）
- Y轴（横轴）
- 矩形块
- 图例

条形图矩形块高度一定，宽度代表数值大小。有多列数据映射到Y轴时，采用分组柱状形式显示。

配置项

表 8-3: 配置项说明

配置项	说明
X轴	一般为分类数据。
Y轴	可以配置一列或多列数据对应到左轴数值区间。
图例位置	图例在图表中的位置，可以配置为上、下、左和右。
间距	坐标轴距离图表边界距离。

使用步骤

1. 键入查询语句，选择时间区间后点击右侧**查询**按钮。
2. 选择条形图 。
3. 配置图表属性。

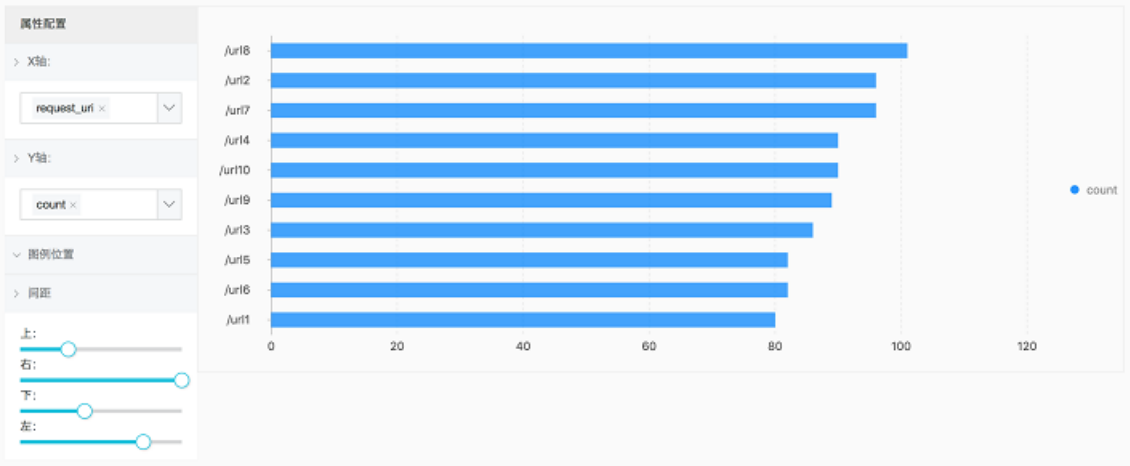
> **说明：**
>
> - 条形图适用于不超过20条的数据，建议使用LIMIT进行控制，以免纵向高度过高导致分析对比情况不直观，分析top场景时候使用ORDER BY配合。同时，当有多列数据映射到Y轴时，建议不要超过5个。
> - 支持使用分组条形图，但是条形图仅适用于同增同减的分类。

简单条形图示例

分析前十访问的`request_uri`：

```
* | select  request_uri, count(1) as count group by request_uri order
by count desc limit 10
```

图 **8-16:** 简单条形图



8.1.10 饼图

饼图用于表示不同分类的占比情况，通过弧度大小来对比各种分类。饼图通过将一个圆饼按照分类的占比划分成多个区块，整个圆饼代表数据的总量，每个区块（圆弧）表示该分类占总体的比例大小，所有区块（圆弧）的加和等于100%。

构成

- 扇形
- 文本百分比
- 图例

配置项

配置项	说明
分类	分类数据。
数值列	分类数据对应的数值。
图例位置	图例在图表中的位置，可以配置为上、下、左和右。
间距	坐标轴距离图表边界距离。

配置项	说明
饼图类型	提供饼图（默认）、环图以及南丁格尔玫瑰图。

类型

日志服务提供默认饼图、环图及南丁格尔玫瑰图三种类型的饼图。

环图

环图本质上是将饼图中心挖空，相比于饼图来说有如下优点：

- 在原有构成的基础上增加了总数显示，展示了更多的信息。
- 两个饼图直接进行比较是非常不直观的，两个环图间可以通过环状条长度进行简单的对比。

南丁格尔玫瑰图

南丁格尔玫瑰图本质上并不是环图，而是在极坐标系下画出来的柱状图，每一个分类数据被圆弧平分，使用圆弧的半径长短表示数据的大小，相比于饼图来说有如下优点：

- 饼图适用于不超过10条的分类数据，南丁格尔玫瑰图则适用于分类较多的场景（10-30条数据）。
- 由于半径和面积是成平方的关系，南丁格尔玫瑰图放大了各个分类数据之间值的差异，尤其适合对比大小相近的数值。
- 由于圆形有周期的特性，南丁格尔玫瑰图也适用于表示一个周期的时间概念，比如星期、月份。

使用步骤

1. 键入查询分析语句，选择时间区间后点击右侧**查询**按钮。
2. 选择饼图 。
3. 配置图表属性。

> 说明：
>
> - 饼图和环图适用于10条以内的数据，建议使用LIMIT进行控制，以免不同色的分面太多导致分析不直观。
> - 分析超过10条数据建议采用南丁格尔玫瑰图或者柱状图。

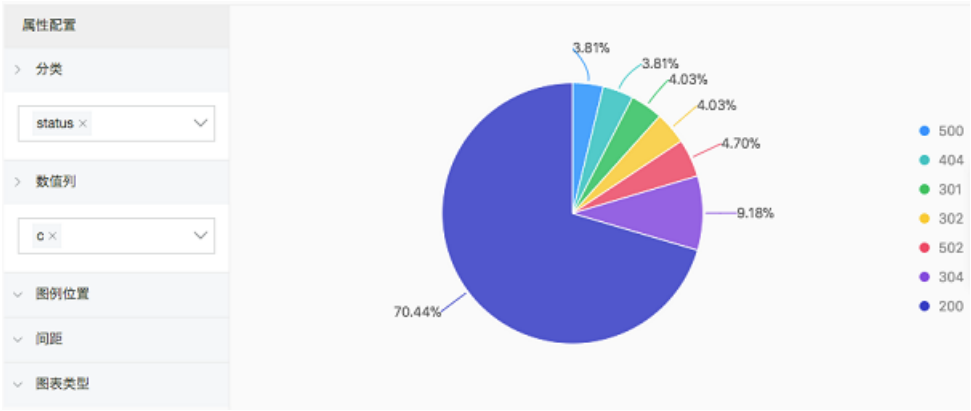
示例

饼图

分析访问status的占比情况：

```
* | select status, count(1) as c group by status order by c limit 10
```

图 **8-17:** 饼图

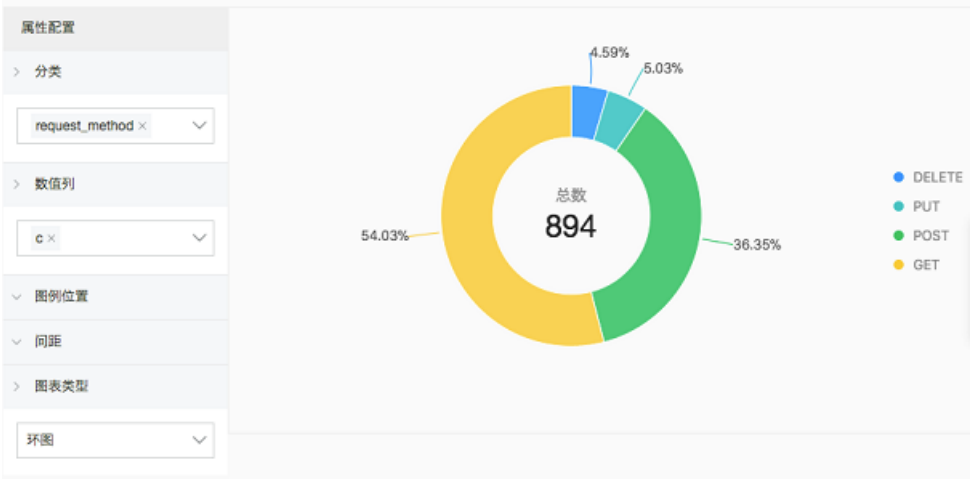


环图

分析访问request_method的占比情况：

```
* | select request_method, count(1) as c group by request_method order
 by c limit 10
```

图 **8-18:** 环图

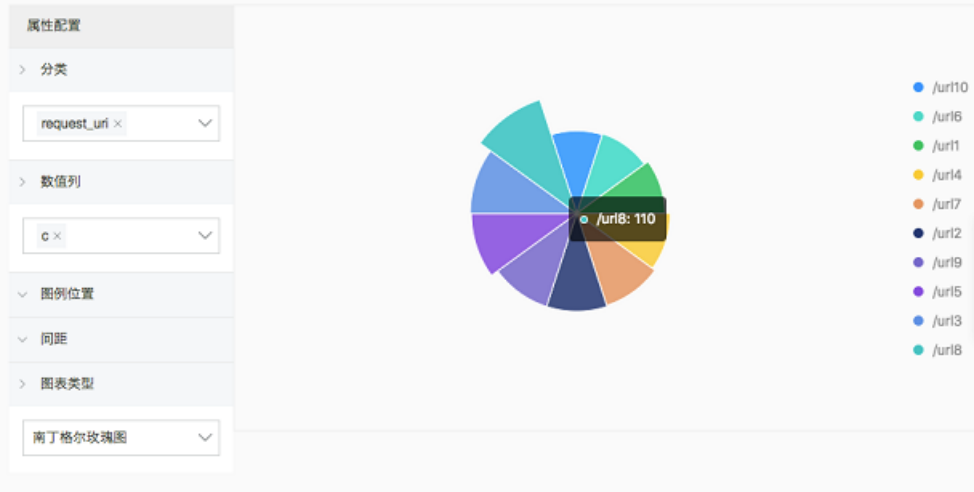


南丁格尔玫瑰图

分析访问request_uri的占比情况：

```
* | select request_uri, count(1) as c group by request_uri order by c
```

图 8-19: 南丁格尔玫瑰图



8.1.11 单值图

单值图可以用于突出显示单个数值。单值图的类型包括：

- 矩形框：用于展示一般数值。
- 刻度盘：用于查看数值与设定阈值的接近程度。
- 同比环比图：用于查看同比和环比函数的SQL查询结果，分析语法请参考同比和环比函数。

默认选择矩形框图显示。矩形框图作为最简单直接的数据表现形式，直观清晰地将某一个点上的数据展示出来，一般用于表示某一个时间点上的关键信息。针对比例类指标的显示，可选用刻度盘类型。

构成

- 主文案
- 单位（可选）
- 描述（可选）
- 分类

配置项

- 矩形框配置说明：

矩形框配置	说明
图表类型	矩形框
数值列	默认选择该列的第一行数据进行展示。
文案	文案相关的属性配置，包括： • 字号（12px-100px） • 单位 • 单位字号（12px-100px） • 数值描述 • 数值描述字号（12px-100px）
颜色	图表中的颜色，包括： • 字体颜色 • 背景颜色

- 刻度盘配置说明：

刻度盘配置类别	配置项	说明
图表类型	刻度盘	将查询结果以刻度盘形式展示。
数值列	实际值	默认选择该列的第一行数据进行展示。
	单位	刻度盘数值的单位。
	刻度盘最大值	刻度盘显示刻度的最大值，默认为100。
	颜色区域个数	即将刻度盘分为几个数值区域，每个区域以不同颜色表示。 颜色区域个数取值范围为5，默认为3个颜色区域。
	区域最大值	刻度盘数值区域的最大值，最后一个区域最大值默认为刻度盘最大值，不需要指定。 **说明：** 刻度盘默认3个颜色区域，且区域范围默认均分，如果您调整了**颜色区域个数**，不会改变

刻度盘配置类别	配置项	说明
		默认区域的范围，您需要根据需求重新指定每个区域的最大值。
	显示标题	您可以在仪表盘中添加刻度盘类型的单值图，**显示标题**用来控制刻度盘形式的单值图标题在仪表盘页面的显示或隐藏。默认为关闭状态，即不显示刻度盘标题。 单击开启后不会在当前页面中显示效果，需要创建或修改报表后在仪表盘页面中查看。
文案	字号	文案的字号，取值范围为12px~100px。
	数值描述	数值的描述内容。
	描述内容字号	描述内容的字号，取值范围为12px~100px。
颜色	区域颜色	默认3个区域，对应的颜色分别为蓝、黄和红。 如果您将**颜色区域个数**更改为3个以上，新增的区域默认为蓝色，您可以重新调整各个区域的颜色。
	字体颜色	数值在仪表盘中显示的颜色。

- 同比环比图配置说明：

刻度盘配置类别	配置项	说明
图表类型	同比环比图	将查询结果以同比环比图形式展示。
数值列	显示值	显示在同比环比图中心的数值，一般设置为同比环比函数中当前时段的统计结果。
	对比值	用于和阈值比较的数值，一般设置为同比环比函数中当前时段和之前时段的对比结果。
	趋势比较阈值	用于衡量对比值变化趋势的数值。
文案	字号	显示值的字号，取值范围为12px~100px。
	单位	显示值的单位。
	单位字号	显示值单位的字号，取值范围为12px~100px。
	比较值单位	比较值的单位。

刻度盘配置类别	配置项	说明
	比较值字号	比较值及其单位的字号，取值范围为12px~100px。
	数值描述	对显示的数值及增长趋势的描述，显示在数值下方。
	描述内容字号	数值描述的字号，取值范围为12px~100px。
颜色	字体颜色	显示值的字体颜色。
	增长字体颜色	对比值大于阈值时，对比值显示的字体颜色。
	增长背景颜色	对比值大于阈值时，显示的背景颜色。
	下降字体颜色	对比值小于阈值时，对比值显示的字体颜色
	下降背景颜色	对比值小于阈值时，显示的背景颜色。
	相等背景颜色	对比值等于阈值时，显示的背景颜色。

使用步骤

1. 键入查询语句，选择时间区间后点击右侧**查询**按钮。
2. 选择单值图123。
3. 根据需求选择图表类型，并配置图表属性。

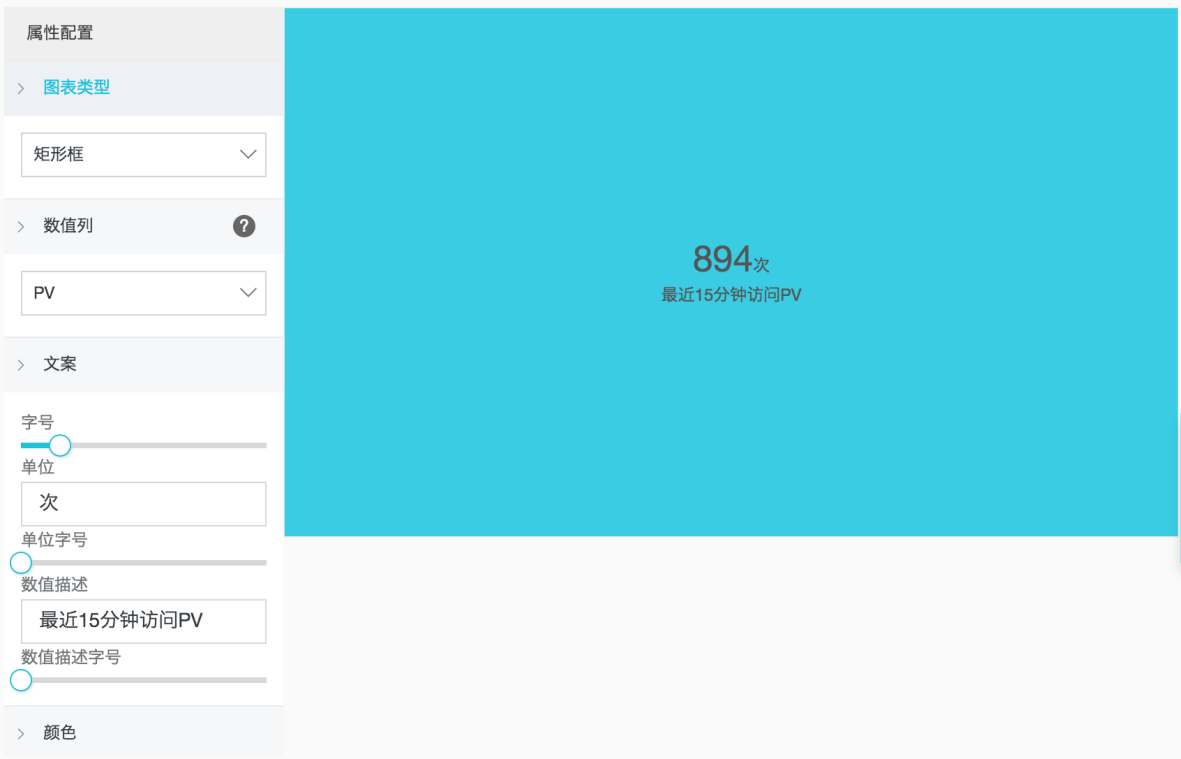
> 说明：
> 日志服务数字图会自动根据数值大小进行归一化操作，如`230000`会被处理为`230K`，如果需要自己定义数值格式，请通过数学计算函数在实时分析阶段进行处理。

示例

执行以下查询分析语句查看访问量，并以图表方式展示分析结果：

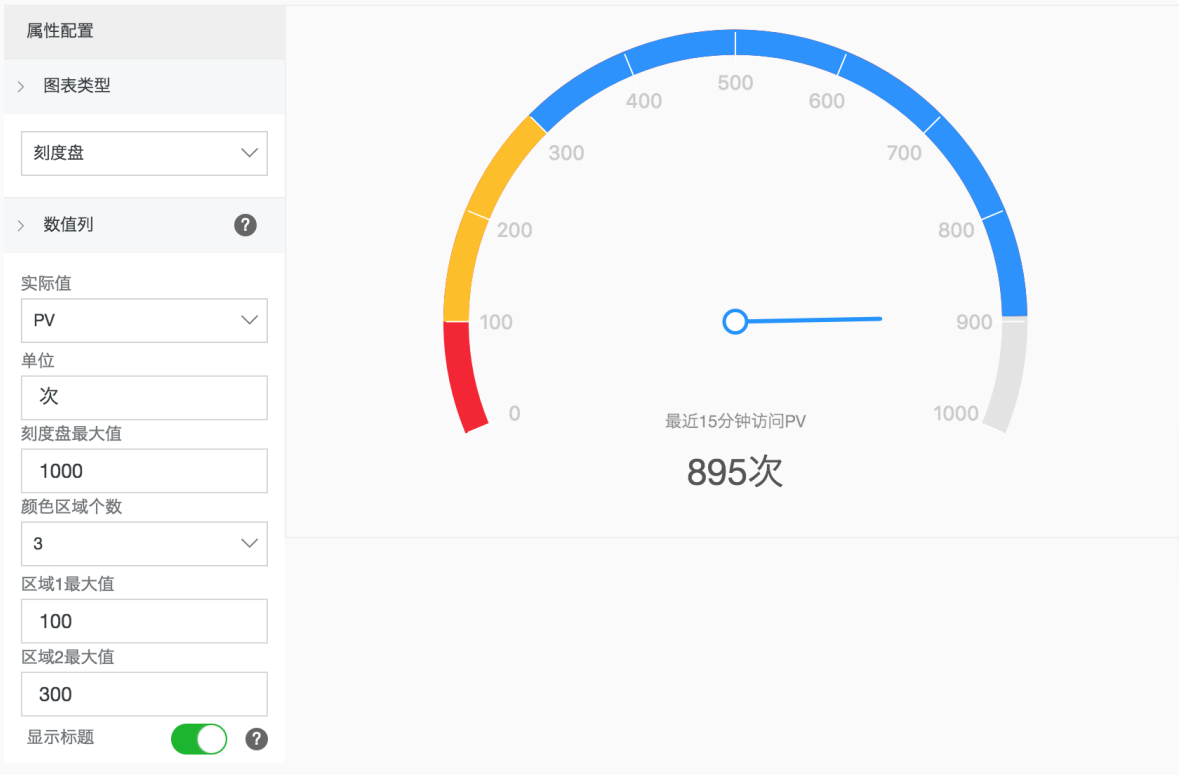
- 矩形框

```
* | select count(1) as pv
```



- 刻度盘

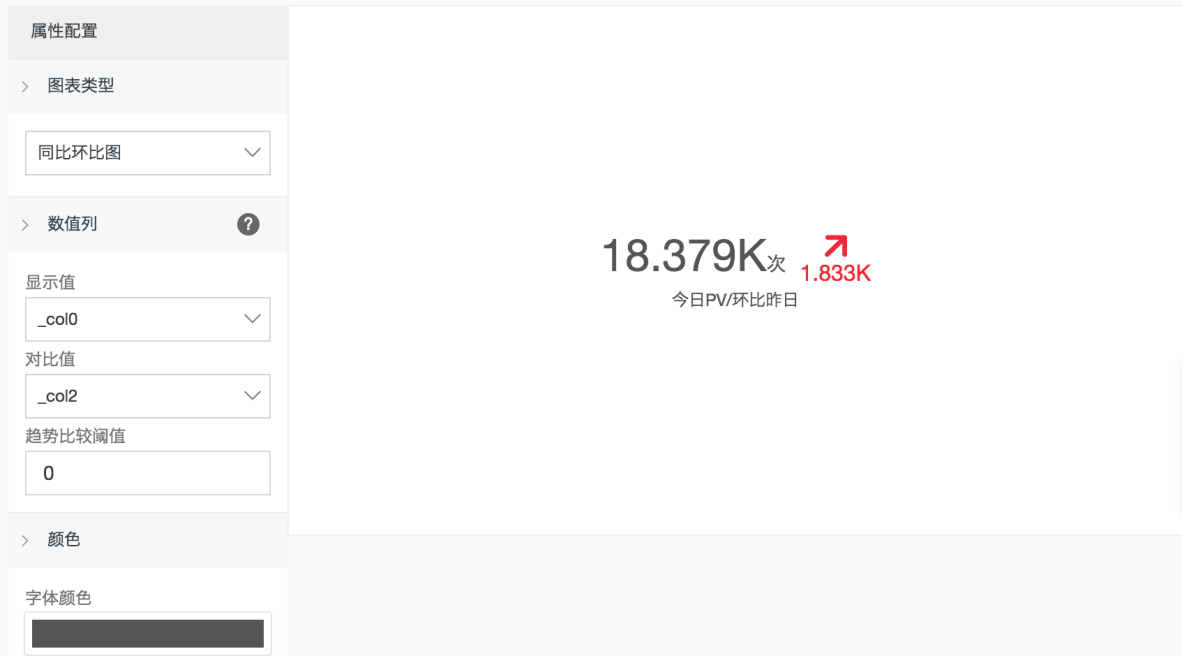
```
* | select count(1) as pv
```



- 同比环比图

查看今天与昨天访问量的对比：

```
* | select diff[1],diff[2], diff[1]-diff[2] from (select compare( pv
 , 86400) as diff from (select count(1) as pv from log))
```



8.1.12 矩形树图

矩形树图，即矩形式树状结构图（Treemap），用矩形面积表示数据的大小。各个小矩形的面积越大，表示占比越大。

基本构成

计算排列得到的矩形块。

配置项

配置	说明
分类	表示数据类别的字段。
数值列	数值字段，某个类别对应的数值越大，其矩形框越大。
间距	矩形框之间上下左右的间距，取值范围为0~100px。

使用步骤

1. 键入查询语句，选择时间区间后单击右侧的**查询**。

2. 选择矩形树图  。

3. 配置图表属性。

示例

分析NGINX日志中hostname分布。

```
* | select hostname, count(1) as count group by hostname order by
count desc limit 1000
```

设置分类为`hostname`，数值列为`count`。



8.1.13 地图

以地图作为背景，通过图形颜色、图像标记的方式展示地理数据信息。日志服务提供了三种地图方式，分别为：中国地图、世界地图以及高德地图（高德地图分为点图和热力图。您可以在查询分析语句中使用特定的函数，日志服务会将您的分析结果以地图方式展示出来。

基本构成

- 地图画布
- 色块

配置项

配置项	说明
位置信息	日志数据中记录的位置信息，在不同的地图类型中以不同的尺度表示。

配置项	说明
	• 省份（中国地图） • 国家（世界地图） • 经纬度（高德地图）
数值列	位置信息对应的数据量。

使用步骤

1. 键入查询语句，选择时间区间后点击右侧**查询**按钮。
   - 中国地图：使用`ip_to_province`函数。
   - 世界地图：使用`ip_to_country`函数。
   - 高德地图：使用`ip_to_geo`函数。
2. 选择地图  。
3. 进行图表属性配置。

应用场景

中国地图

支持使用`ip_to_province`函数生成中国地图。

- **SQL语句**：

```
* | select  ip_to_province(remote_addr) as address, count(1) as count
group by address order by count desc limit 10
```

- **数据集**：

address	count
广东省	163
浙江省	110
福建省	107
北京市	89
重庆市	28
黑龙江省	19

- 省份信息选择`address`，数值列选择`count`。

图 8-20: 中国地图



世界地图

支持使用ip_to_country函数生成世界地图。

- SQL语句：

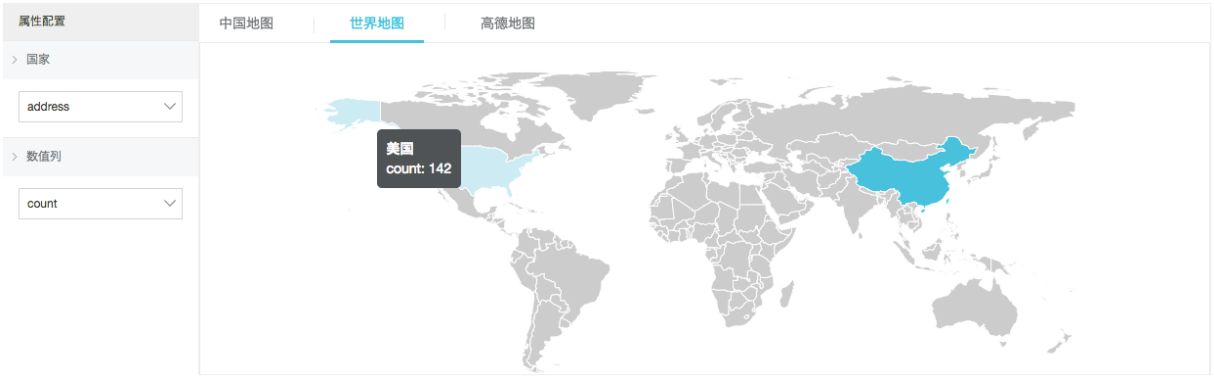
```
* | select  ip_to_country(remote_addr) as address, count(1) as count
 group by address order by count desc limit 10
```

- 数据集：

address	count
中国	8354
美国	142

- 国家信息选择address，数值列选择count。

图 8-21: 世界地图



高德地图

支持使用ip_to_geo函数生成高德地图。数据集先纬后经，以“，”为分隔符，如果数据为两列lng（经度）和lat（纬度），可以使用concat('lat', ',', lng')合并为一列。

- **SQL语句**：

```
* | select  ip_to_geo(remote_addr) as address, count(1) as count
group by address order by count desc limit 10
```

- 数据集：

address	count
39.9289,116.388	771
39.1422,117.177	724
29.5628,106.553	651
30.2936,120.161420	577
26.0614,119.306	545
34.2583,108.929	486

- 经纬度信息选择address，数值列选择count。

图 **8-22:** 高德地图-点图



默认返回点图。如数据点分布密集，您也可以切换为热力图。

图 8-23: 高德地图-热力图



8.1.14 面积图

面积图是在折线图的基础之上形成的，它将折线图中折线与坐标轴之间的区域使用颜色进行填充，这个填充即为我们所说的面积，颜色的填充可以更好的突出趋势信息。和折线图一样，面积图强调数量随时间而变化的程度，用于突出总值趋势。它们最常用于表现趋势和关系，而不是传达特定的值。

基本构成

- X轴（横轴）
- Y轴（纵轴）
- 面积块

配置项

配置项	说明
X轴	一般为有序数据类别（时间序列）。
Y轴	可以配置一列或多列数据对应到左轴数值区间。
图例位置	图例在图表中的位置，可以配置为上、下、左和右。
间距	坐标轴距离图表边界距离

使用步骤

1. 键入查询语句，选择时间区间后点击右侧**查询**按钮。
2. 选择面积图 。

3. 配置图表属性。

> 说明：
> 面积图单个面积块数据记录数要大于2，以免无法分析数据趋势，同时，建议同一个图上不要超过5组面积块。

示例

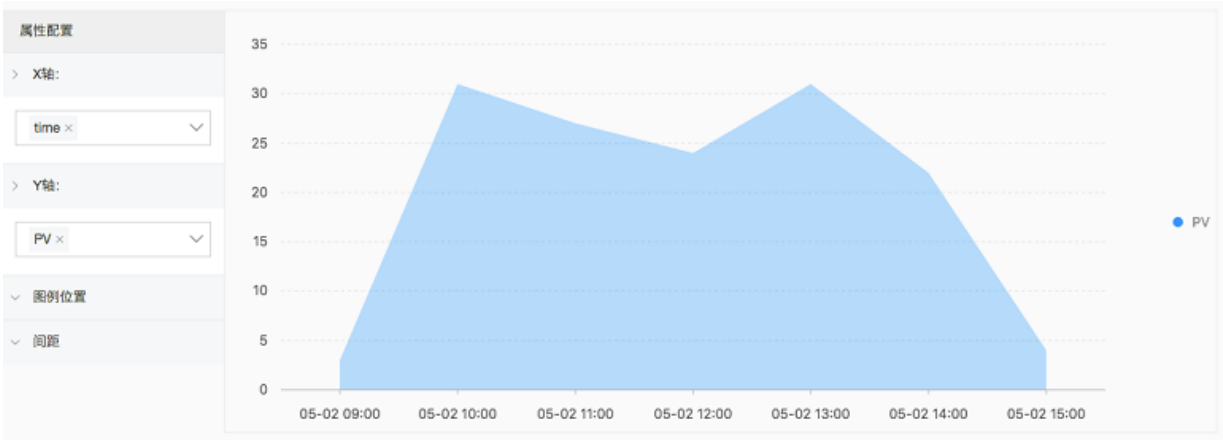
简单面图

`42.0.192.0`这个IP在最近1天内的访问情况：

```
remote_addr: 42.0.192.0 | select date_format(date_trunc('hour',
__time__), '%m-%d %H:%i') as time, count(1) as PV group by time order
by time limit 1000
```

X轴选择`time`，Y轴选择`PV`。

图 **8-24:** 简单面图

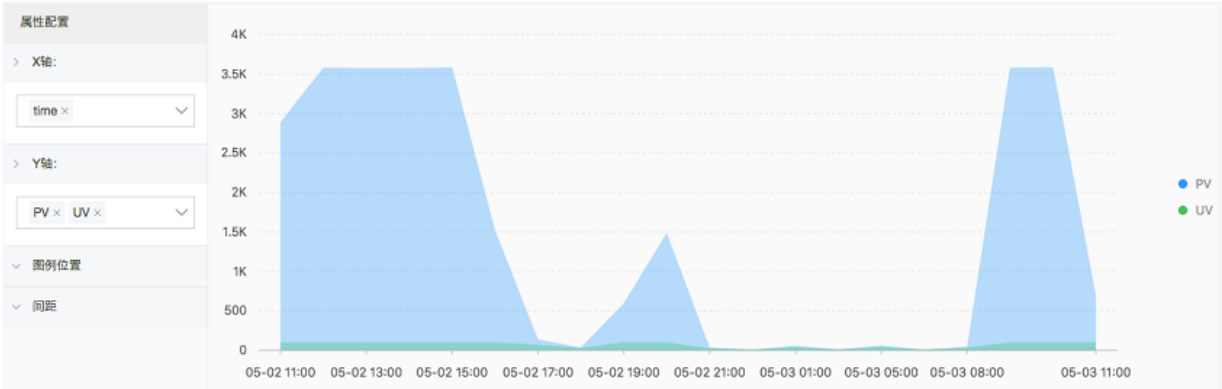


层叠面图

```
* | select date_format(date_trunc('hour', __time__), '%m-%d %H:%i')
as time, count(1) as PV, approx_distinct(remote_addr) as UV group by
time order by time limit 1000
```

X轴选择`time`，Y轴选择`PV`和`UV`。

图 8-25: 层叠面图



8.1.15 流图

流图（Flow Chart）也叫主题河流图（ThemeRiver），是围绕中心轴线进行布局的一种堆叠面积图。不同颜色的条带状分支代表了不同的分类信息，条状带的宽度映射了对应的数值大小。此外，原数据集中的时间属性，映射到X轴上，是一个三维关系的展现。

流图可以通过图表类型切换为线图和柱状图，需要注意的是柱状图默认以层叠形式展现，不同分类数据的起点是从上个柱状的顶部开始。

基本构成

- X轴（横轴）
- Y轴（纵轴）
- 条状

配置项

配置项	说明
X轴	一般为有序数据类别（时间序列）。
Y轴	可以配置一列或多列数据对应到左轴数值区间。
聚合列	需要在第三维上聚合的信息。
图例位置	图例在图表中的位置，可以配置为上、下、左和右。
间距	坐标轴距离图表边界距离。
图表类型	提供面图（默认）、线图以及柱状图（层叠）。

使用步骤

1. 键入查询语句，选择时间区间后点击右侧**查询**按钮。
2. 选择  ，即流图。
3. 配置图表属性。

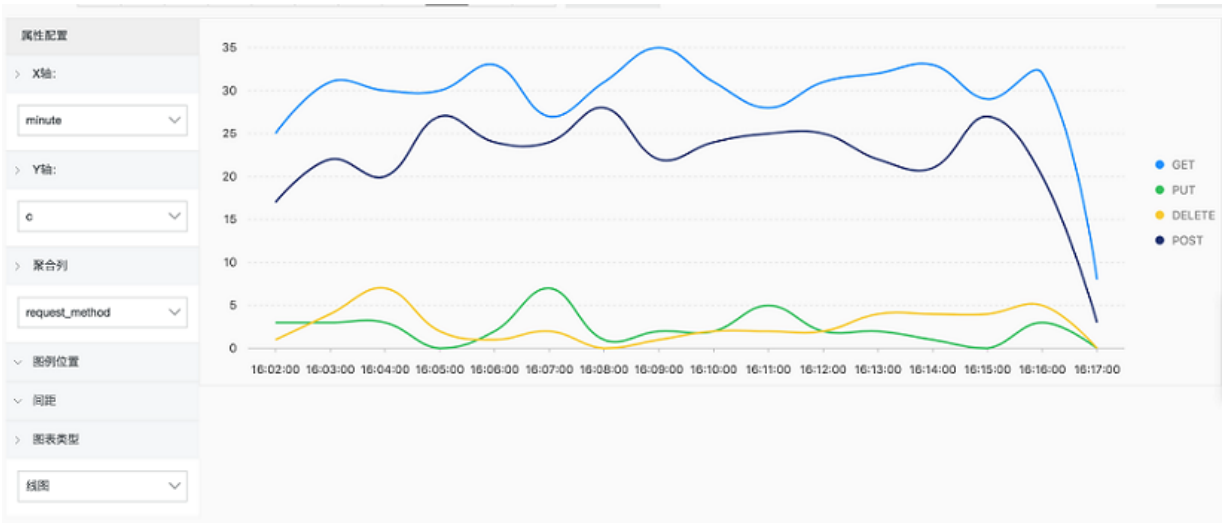
示例

流图适合三维关系的展示，时间-分类-数值的展现。

```
* | select date_format(from_unixtime(__time__ - __time__% 60), '%H:
%i:%S') as minute, count(1) as c,  request_method group by minute,
request_method order by minute asc limit 100000
```

X轴选择`minute`，Y轴选择`c`，聚合列选择`request_method`。

图 **8-26:** 流图



8.1.16 桑基图

桑基图 (Sankey Diagram)，是一种特定类型的流图，用于描述一组值到另一组值的流向。适合网络流量等场景，通常包含3组值`source`、`target`以及`value`。`source`和`target`描述了节点的关系，而`value`描述了该`source`和`target`之间边的关系。

功能特点

桑基图具有以下特点：

- 起始流量和结束流量相同，所有主支宽度的总和与所有分出去的分支宽度总和相等，保持能量的平衡。

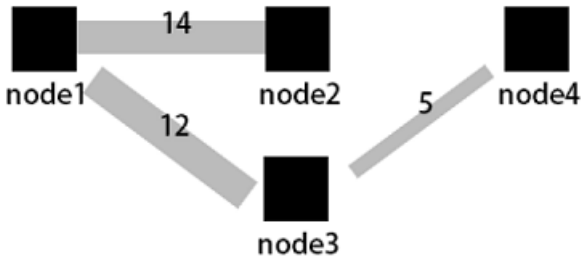
- 在内部，不同的线条代表了不同的流量分流情况，它的宽度成比例地显示此分支占有的流量。
- 节点不同的宽度代表了特定状态下的流量大小。

例如以下数据可以用桑基图表示：

source	target	value
node1	node2	14
node1	node3	12
node3	node4	5
…	..	…

桑基图如此描述上述数据的关系：

图 8-27: 桑基图的数据关系



基本构成

- 节点
- 边

配置项

配置项	说明
起点列	描述起始节点。
终点列	描述终点节点。
数值列	链接起点节点和终点节点的值。
间距	坐标轴距离图表边界距离。

使用步骤

1. 键入查询语句，选择时间区间后点击右侧**查询**按钮。
2. 选择桑基图 。
3. 配置图表属性。

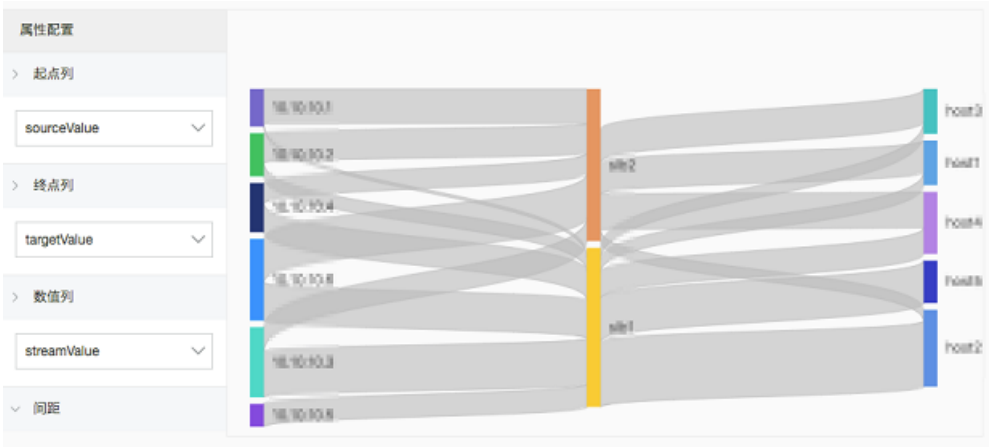
示例

普通桑基图

如果日志字段包含了`source`、`target`和`value`，即每条日志本身就是节点和边的关系，可以通过嵌套子查询获取到`steamValue`的总和。

```
* | select sourceValue, targetValue, sum(streamValue) as streamValue
from (select sourceValue, targetValue,
 streamValue, __time__ from log group by  sourceValue, targetValue,
streamValue, __time__ order by __time__ desc) group by sourceValue,
 targetValue
```

图 **8-28:** 普通桑基图

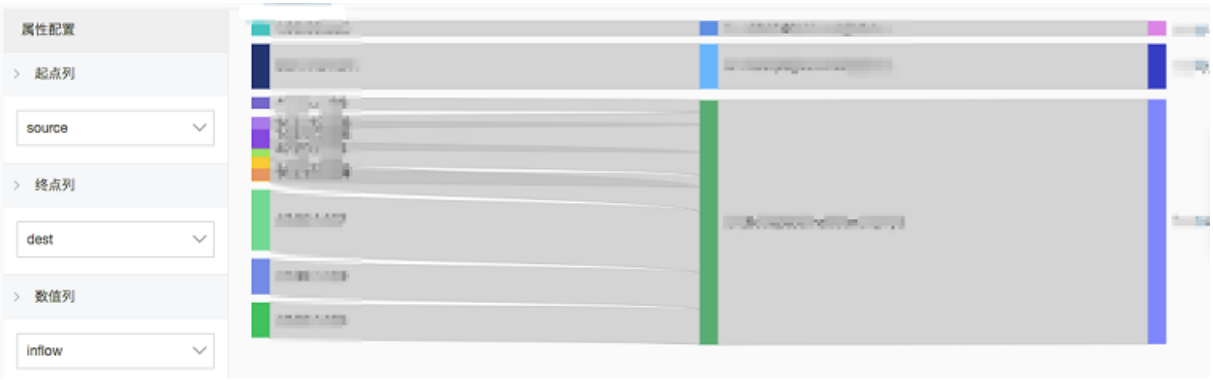


负载均衡7层访问日志场景

日志服务支持负载均衡7层访问日志，可以直接通过访问日志绘制桑基图。

```
* | select COALESCE(client_ip, slbid, host) as source, COALESCE(host
, slbid, client_ip) as dest, sum(request_length) as inflow group by
grouping sets( (client_ip, slbid), (slbid, host))
```

图 8-29: 嵌套子查询



8.1.17 词云

词云，是文本数据的视觉表示，由词汇组成类似云的彩色图形，用于展示大量文本数据。每个词的重要性以字体大小或颜色显示，能最让用户最快速地感知某一些关键词的权重大小。

基本构成

词云类型的图表为您展示经过数据计算排列的词。

配置项

配置项	说明
词列	代表要展示的一组词的信息。
数值列	每一个词对应的数值信息。
字号	合理调整字号范围以适应画布。 • 最大字号（50px-80px） • 最小字号（12px-24px）
间距	坐标轴距离图表边界距离

使用步骤

1. 键入查询语句，选择时间区间后点击右侧**查询**按钮。
2. 选择词云 。
3. 配置图表属性。

示例

分析NGINX日志中hostname分布：

```
* | select hostname, count(1) as count group by hostname order by
count desc limit 1000
```

词列为`hostname`，数值列为`count`。

图 **8-30:** 词云



9 告警

9.1 简介

日志服务支持根据快速查询设置告警，实现实时的服务状态监控。

功能实现

日志服务的告警功能基于快速查询实现。在日志服务控制台**查询页面**设置告警规则，并指定告警规则的属性、检查条件和告警动作。设置告警后，日志服务定期对采集到的日志数据自动执行快速查询，查询结果满足预设条件时发送告警通知，实现实时的服务状态监控。

- **告警规则属性**：可设置数据检查的时间和间隔，日志服务根据设置定期执行检查。
- **检查条件**：可设置检查条件的比较字段、比较符和检查阈值。比较符和检查阈值构成检查条件，比较字段的查询结果满足检查条件时发送告警通知。
- **告警动作**：设置告警的通知方式和内容。推荐设置通知方式为**通知中心**，告警通知会以站内通知和短信、邮件的形式发送给通知中心预设的联系人。

告警中的快速查询

快速查询中的查询语句可以是查询语句或查询分析语句。

- 查询语句：直接返回查询条件命中的日志数据。
- 查询分析语句：对查询条件命中的日志进行统计，返回统计结果。
- **查询语句**

  例如，查询最近 15 分钟内包含 error 的数据，条件为 error，一共有 154 条。每条内容都是 Key、Value 组合，您可以对某个 Key 下的 Value 设置告警条件。

  > **说明：**
  > 对于查询结果一次超过 10 条的情况，告警规则只判断前10条 ，只有前10条中任意一条符合条件，才会触发告警。

图 9-1: 查询语句



- 查询分析语句

例如查询所有日志中状态码为200的日志比例，查询语句如下（查询语法请参考查询语法）：

```
* | select sum(case when status=200 then 1 else 0 end) *1.0/count(1)
  as ratio
```

因此，可以设置告警检查条件为`ratio < 0.9`，表示当状态码为200的日志小于总日志数的90%时进行告警。

图 9-2: 查询分析语句



9.2 设置告警

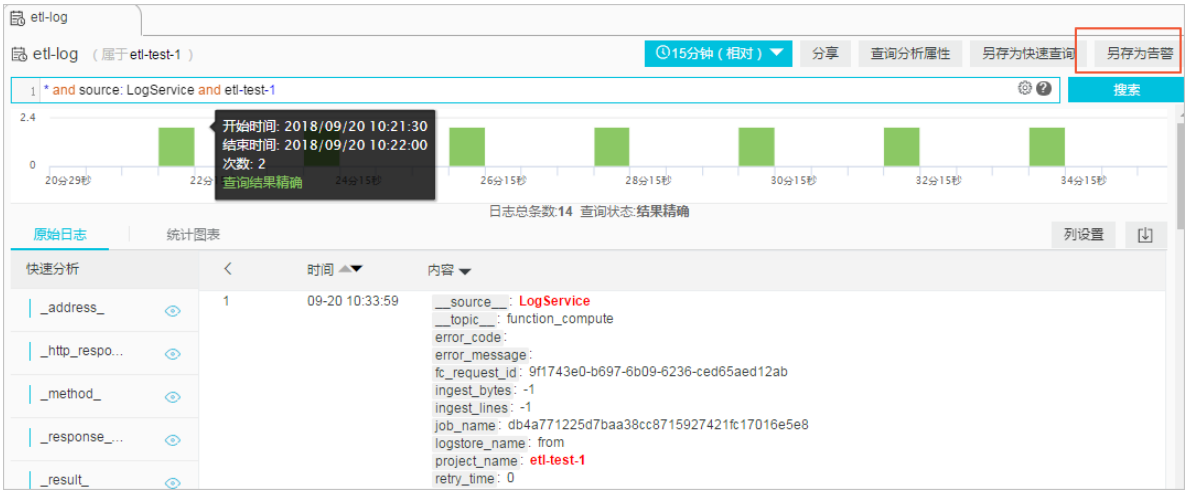
在查询页面将快速查询另存为告警，日志服务会定时执行检查，并在满足告警条件时发送告警信息。

前提条件

- 已采集到日志数据。
- 已开启并配置索引。

操作步骤

1. 登录日志服务控制台，单击Project名称。
2. 在**Logstore**列表页面，单击查询分析列下的 查询。
3. 在搜索框中输入查询语句。
4. 设置查询的时间范围，并单击搜索。
5. 在页面右上角单击另存为告警。



6. 设置告警规则并单击 确定。

规则	说明
告警规则名称	告警规则的名称。名称长度为3~63个字符。
告警规则属性	
快速查询名称	设置告警中使用的快速查询名称，可以设置为： • 使用当前查询，表示使用当前的查询语句作为快速查询。 • 已存在的其他快速查询。创建快速查询请参考快速查询。

规则	说明
数据查询时间（分钟）	服务端每次执行告警检查读取的数据时间范围。例如，设置为1时表示即对最近1分钟到执行检查的当前时间进行查询。 数据查询时间单位为分钟，取值范围为1~60。 **说明：** 目前服务端每次告警规则检查只会采样处理时间区间开始的前 10 条数据。
检查间隔（分钟）	服务端每次执行告警检查的时间间隔。 检查间隔单位为分钟，取值范围为1~1,440。
触发次数	告警检查连续触发的次数，达到指定的触发次数后立即发送告警通知。 取值范围为1~10,000。 例如，检查间隔（分钟）设置为 1，触发次数设置为2，表示每隔1分钟执行一次检查，连续 2 次检查均满足告警条件即发送告警，连续两次告警通知之间最小间隔为 2 分钟。
检查条件	
字段名称	日志内容中用于告警的 Key 名称。
比较符	检查条件中的比较符，支持数值类和字符类比较符。比较符说明请参考比较符。
检查阈值	检查条件中的比较值，与比较符组合判断快速查询的结果是否符合告警条件。
告警动作	
通知类型	发送告警通知的方式。当您设置的告警规则被触发后，日志服务将通过您预先设定的通知方式，为您发送告警提醒。 当前支持以下告警通知方式，详细说明请查看通知方式。 • 通知中心（推荐）：向阿里云通知中心设置的联系人发送短信或邮件。 • 短信（即将下线）：向预设的手机号码发送告警通知。 • **WebHook-钉钉机器人**：在钉钉群中以钉钉机器人消息的方式发送通知。 • **WebHook-自定义**：以Post方式发送通知到自定义WebHook链接。
手机号码	接收告警通知的手机号码。通知类型为短信时配置。

规则	说明
WebHook地址	WebHook的URL地址。通知类型为**WebHook-**钉钉机器人和**WebHook-**自定义时配置。
通知内容	告警通知的内容。最多支持500个字符。

表 **9-1:** 比较符

操作	描述	示例
>	该列是否大于数值	$count > 0
<	该列是否小于数值	$count<200
>=	大于等于某个数值	$count>=0
<=	小于等于某个数值	$count<=0
like	匹配子串	$project like “admin”
regex	正则匹配字符串	$project regex match “^/S+$”

图 9-3: 报警规则



预期结果

创建完成告警规则后，您可以查看具体的告警记录。

1. 在**Logstore**列表页面，选择左侧导航栏中的**Search/Analytics - 查询分析 > 告警配置**。
2. 选择要查看的告警规则并单击右侧的**查看告警记录**即可查看具体告警记录即对应状态。

告警状态 ：

- **成功**：规则被成功执行，在**告警触发详情**中显示触发标准。
- **失败**：查询、告警规则、或通知阶段失败，可以查看**告警触发详情** 获得详细信息。

- 查询失败：一般为语法不正确。
- 查询调用失败：请提交工单。
- 规则调用失败：请检查规则参数和返回数据格式是否一致。

9.3 通知方式

日志服务的告警功能支持多种通知方式。

包括：

- 通知中心#推荐#
- 短信#即将下线#
- *WebHook*-钉钉机器人
- *WebHook*-自定义

告警信息中包含以下内容：

内容	说明
Uid	阿里云账号ID，即AliUid。
Project	产生告警的日志服务Project名称。
Trigger	告警的名称。
Condition	告警中配置的检查条件。
Message	告警的通知内容。
Context	查询结果。其中，短信和通知中心方式的Context最大长度为100字节，WebHook-钉钉机器人和WebHook-自定义方式为1000字节。

通知中心（推荐）

阿里云消息中心中可设置日志服务告警的联系人，当触发告警时，告警通知会以消息中心中预设的通知方式发送告警通知。

配置步骤

1. 设置告警，且通知类型设置为通知中心。



2. 在阿里云消息中心，单击消息接收管理 > 基本接收管理。



3. 在消息类型 > 日志服务（LOG）告警对应的消息接收人一列单击修改。

图 9-4: 修改消息接收人



4. 在修改消息接收人窗口选择消息接收人。

如您需要新增一位消息接收人，可以直接单击**+新增消息接收人**，并配置该人员用于接收告警信息的邮箱、手机号码和职位信息。仅账号负责人可以为消息接收人配置手机号码。

> **说明：**
>
> - 系统将自动发送验证信息到所填手机号和邮箱，通过验证后方可接收消息。
> - 最少需要设置1位消息接收人。
> - 通知方式默认为邮件 + 短信，且不可更改。
> - 每个手机号或邮箱 一天最多发送**50**次告警通知。

告警示例：

- 短信：



- 邮件：

> **说明：**
>
> 邮件标题为“日志服务告警”。



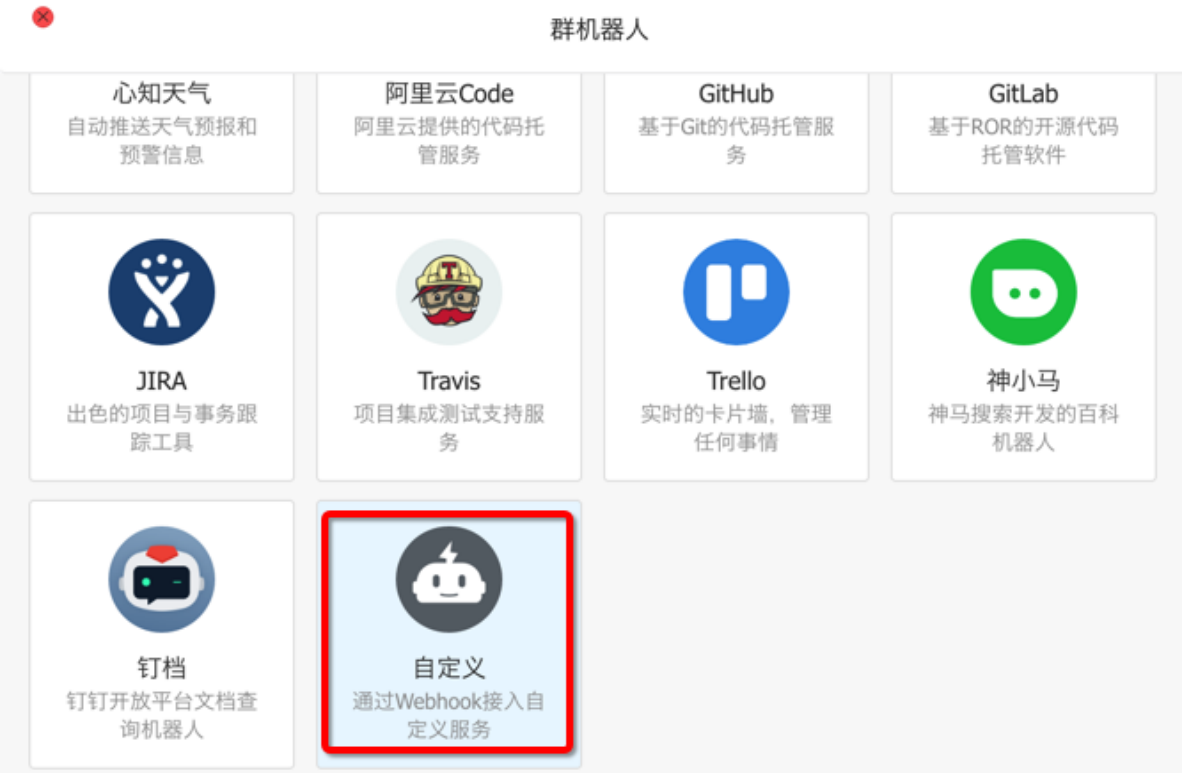
WebHook-钉钉机器人

告警的通知方式可设置为WebHook-钉钉机器人，当触发告警时，告警通知会以钉钉机器人消息的形式发送到钉钉群中。

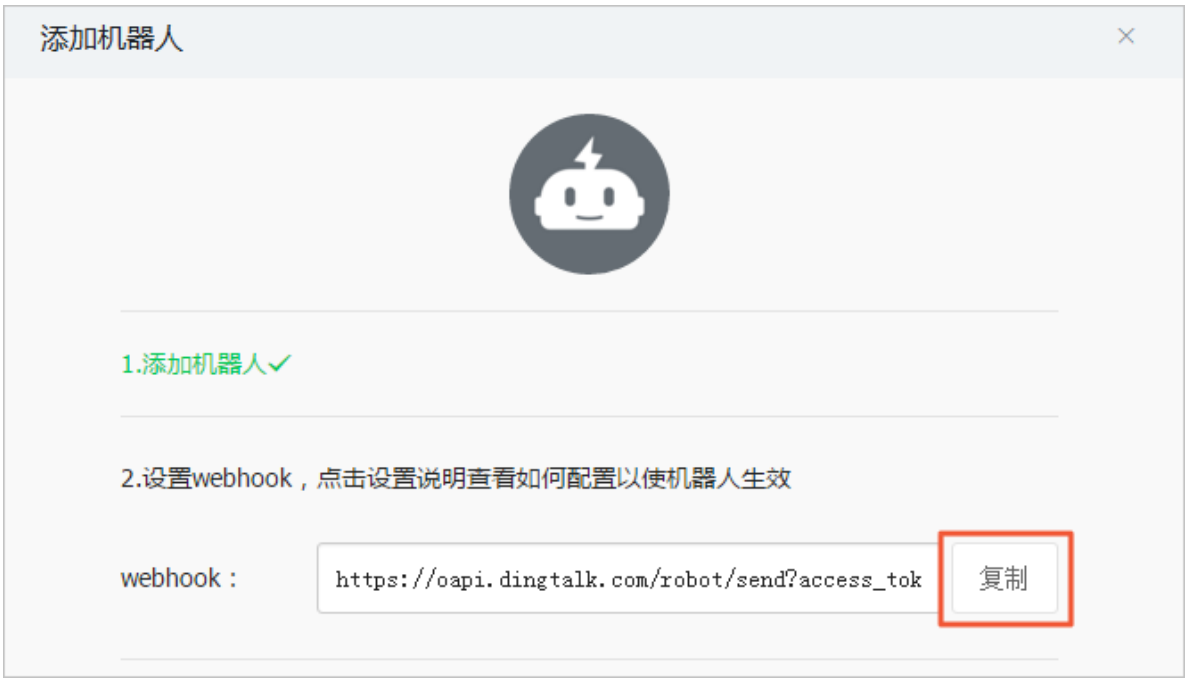
配置步骤

1. 打开钉钉客户端，进入钉钉群。
2. 单击右上角群设置图标，并单击群机器人。
3. 选择自定义（通过**WebHook**接入自定义服务），并单击添加。

图 **9-5:** 钉钉机器人

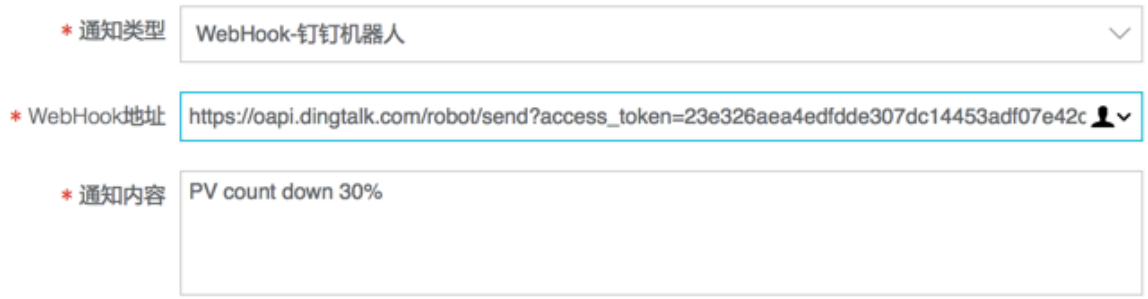


4. 输入机器人名字，并单击完成。
5. 单击复制，复制WebHook链接。



6. 在日志服务控制台设置告警，且通知类型设置为**WebHook-钉钉机器人**。
7. **WebHook地址**一项中，粘贴步骤5中复制的地址，并填写通知内容。

图 **9-6:** 通知内容



告警示例 ：

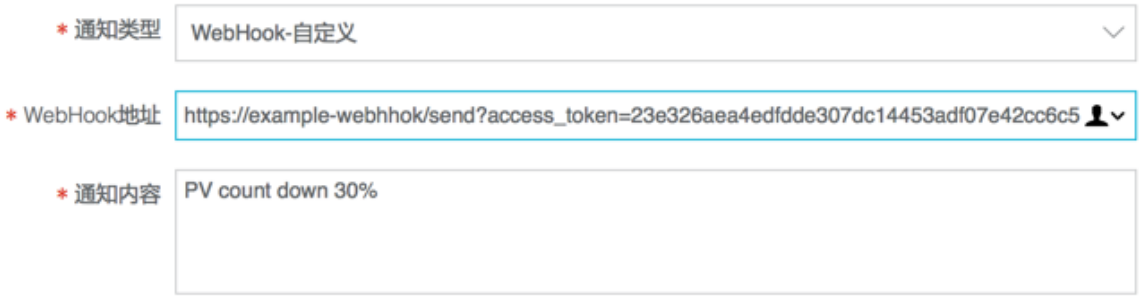


WebHook-自定义

告警的通知方式可设置为WebHook-自定义，当触发告警时，告警通知会以Post方式发送到自定义WebHook地址中。

配置步骤

1. 在日志服务控制台设置告警。
2. 通知类型设置为**WebHook-自定义**。
3. **WebHook**地址中填写自定义的WebHook地址，并填写通知内容。



发生告警后会以Post方式将告警内容发到自定义WebHook地址。

示例：

```
{"uid":
            "13415134513","project":"ali-cn","trigger":"oplog_alert
","condition":"3413 >
            3000", "message":"PV count down 30%",
      "context":"c:3413"}
```

短信通知（即将下线）

告警的通知方式可设置为短信，当触发告警时，日志服务会向预设的手机号码发送短信通知。

配置步骤

1. 在日志服务控制台设置告警。
2. 通知类型设置为短信。
3. 手机号码中填写接收告警通知的短信号码，通知内容中填写告警通知的标题。

   当前，一个告警配置只支持一个手机号码，同一个手机号码一天短信通知上限为 20 条。

10 实时消费

10.1 实时消费简介

数据收集至日志服务 LogHub 后，有三种方法可以处理日志：

方式	场景	实时性	存储时间
实时消费（LogHub）	流计算、实时计算等	实时	自定义
索引查询（LogSearch）	适合最近热数据的在线查询	实时（99.9% 1秒，最大3秒)	自定义
投递存储（LogShipper）	适合全量存储日志，进行离线分析	5~30分钟	依赖于存储系统

实时消费

在写入日志后，最基本功能就是如何消费日志（消费日志与查询日志都意味着“读取”日志）。对于一个 Shard 中日志，消费过程如下：

1. 根据时间、Begin、End 等条件获得游标。
2. 通过游标，多次参数读取日志，同时返回下一个位置游标。
3. 下次移动游标进行日志消费。

除最基本的 API 外，日志服务提供 SDK、Storm Spout、Spark Client、Web Console 等方式进行日志消费：

- 使用*Spark Streaming Client*。
- 使用*Storm Spout*。
- 使用*Flink Connector*：包含Flink Consumer & Producer。
- 使用*LogHub Consumer Library*是对LogHub消费者提供的高级模式。它提供了一个轻量级计算框架，解决多个消费者同时消费Logstore时自动分配Shard以及保序的问题，详情请参考*Consumer Library*。
- 使用*SDK*消费日志：日志服务提供多种语言（Java 和 Python）的 SDK，且这些语言的SDK都支持日志消费接口。关于SDK的更多信息请参考 日志服务 *SDK*。
- 使用云产品消费日志：
  - 使用 *CloudMonitor* 云监控消费：监控场景。

- 使用 API/SDK 消费：业务实时监控场景。
- 使用 Storm Spout 消费日志：自定义监控场景。
- 使用 E-MapReduce 消费日志：参见Storm，Spark Streaming。
- 使用命令行工具 CLI 消费日志。

查询分析

参见实时查询分析简介：

- 使用日志服务控制台查询日志：参见实时查询分析简介。
- 使用日志服务 SDK/API 查询日志：日志服务提供 REST 风格的 API，基于 HTTP 协议实现。日志服务的 API 同样提供全功能的日志查询接口。详细内容请参考日志服务 API。

投递存储

- 投递日志到OSS：长时间存储或使用 E-MapReduce 分析日志。
- 投递至表格存储#Table Store#：使用表格存储（NoSQL）存储日志。
- 通过日志服务投递日志到MaxCompute：使用MaxCompute分析日志。
- 使用函数计算进行自定义投递。
- 通过DataWorks投递数据到MaxCompute：将日志数据通过DataWorks的数据集成（Data Integration）功能投递至MaxCompute，已进行大数据分析。

其他

安全日志服务：日志服务与安全云产品对接，可通过 ISV 消费云产品日志。

10.2 普通消费

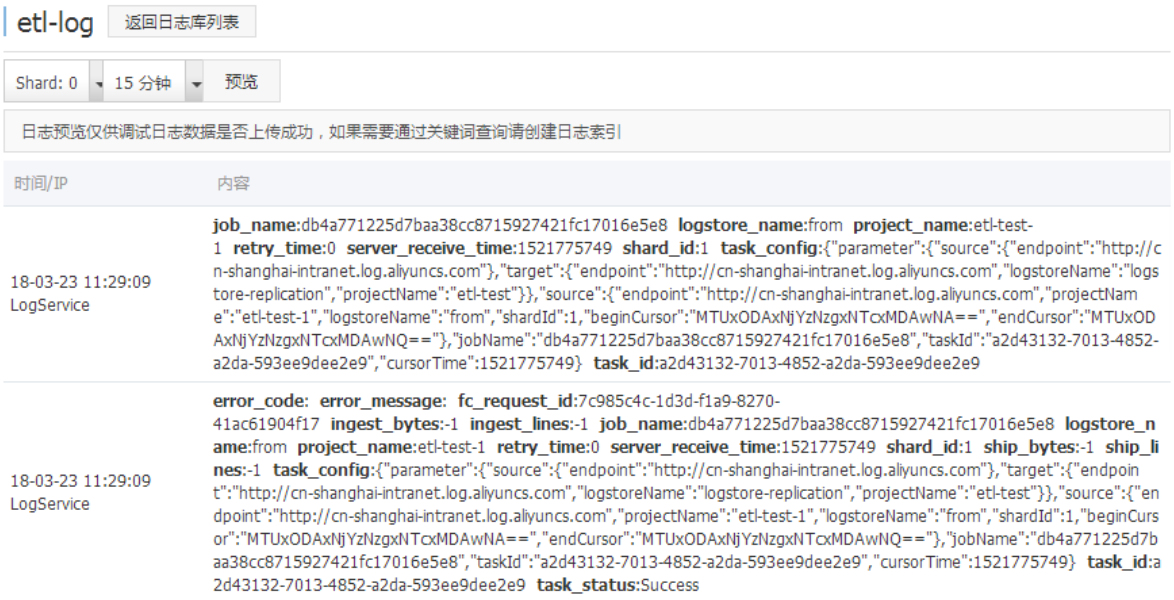
日志预览是普通形式的日志消费。日志服务控制台提供专门的预览页面帮助您在浏览器内直接预览日志库中部分日志。

操作步骤

1. 登录日志服务控制台。
2. 选择所需的项目，单击项目名称进入**Logstore**列表。
3. 在 **Logstore**列表 页面，选择要查看的日志库并单击**预览**。
4. 在日志预览页面，指定预览的 Shard 并指定时间段，然后单击 预览。

   日志预览页面向您展示指定时间区间开始的10个数据包的日志数据。

图 10-1: 普通消费



10.3 消费组消费

10.3.1 消费组消费

协同消费库（Consumer Library）是对日志服务中日志进行消费的高级模式，提供了消费组（ConsumerGroup）的概念对消费端进行抽象和管理，和直接使用SDK进行数据读取的区别在于，用户无需关心日志服务的实现细节，只需要专注于业务逻辑，另外，消费者之间的负载均衡、failover等用户也都无需关心。

Spark Streaming、*Storm* 以及Flink Connector都以Consumer Library作为基础实现。

基本概念

使用消费库之前有两个概念需要理解，分别是消费组（ConsumerGroup）、消费者（Consumer）。

- 消费组

  一个消费组由多个消费者构成，同一个消费组下面的消费者共同消费一个logstore中的数据，消费者之间不会重复消费数据。

- 消费者

  消费组的构成单元，实际承担消费任务，同一个消费组下面的消费者名称必须不同。

在日志服务中，一个logstore下面会有多个shard，协同消费库的功能就是将shard分配给一个消费组下面的消费者，分配方式遵循以下原则：

- 每个shard只会分配到一个消费者。
- 一个消费者可以同时拥有多个shard。

新的消费者加入一个消费组，这个消费组下面的shard从属关系会调整，以达到消费负载均衡的目的，但是上面的分配原则不会变，分配过程对用户透明。

协同消费库的另一个功能是保存checkpoint，方便程序故障恢复时能接着从断点继续消费，从而保证数据不会被重复消费。

使用说明

maven 依赖

```
<dependency>
  <groupId>com.google.protobuf</groupId>
  <artifactId>protobuf-java</artifactId>
  <version>2.5.0</version>
</dependency>
<dependency>
<groupId>com.aliyun.openservices</groupId>
<artifactId>loghub-client-lib</artifactId>
<version>0.6.15</version>
</dependency>
```

main .java文件

```
public class Main {
  // 日志服务域名，根据实际情况填写
  private static String sEndpoint = "cn-hangzhou.log.aliyuncs.com";
  // 日志服务项目名称，根据实际情况填写
  private static String sProject = "ali-cn-hangzhou-sls-admin";
  // 日志库名称，根据实际情况填写
  private static String sLogstore = "sls_operation_log";
  // 消费组名称，根据实际情况填写
  private static String sConsumerGroup = "consumerGroupX";
  // 消费数据的ak，根据实际情况填写
  private static String sAccessKeyId = "";
  private static String sAccessKey = "";
  public static void main(String []args) throws LogHubClientWorkerEx
ception, InterruptedException
  {
        // 第二个参数是消费者名称，同一个消费组下面的消费者名称必须不同，可以使用相同的消费组名称，不同的消费者名称在多台机器上启动多个进程，来均衡消费一个Logstore，这个时候消费者名称可以使用机器ip来区分。第9个参数（maxFetchLogGroupSize）是每次从服务端获取的LogGroup数目，使用默认值即可，如有调整请注意取值范围(0,1000]
      LogHubConfig config = new LogHubConfig(sConsumerGroup, "
consumer_1", sEndpoint, sProject, sLogstore, sAccessKeyId, sAccessKey
, LogHubConfig.ConsumePosition.BEGIN_CURSOR);
```

```
      ClientWorker worker = new ClientWorker(new SampleLogHubProcesso
rFactory(), config);
        Thread thread = new Thread(worker);
        //Thread运行之后，Client Worker会自动运行，ClientWorker扩展了
Runnable接口。
       thread.start();
       Thread.sleep(60 * 60 * 1000);
       //调用worker的Shutdown函数，退出消费实例，关联的线程也会自动停止。
       worker.shutdown();
       //ClientWorker运行过程中会生成多个异步的Task，Shutdown之后最好等待还在
执行的Task安全退出，建议sleep 30s。
       Thread.sleep(30 * 1000);
  }
}
```

SampleLogHubProcessor.java文件

```
public class SampleLogHubProcessor implements ILogHubProcessor
{
  private int mShardId;
  // 记录上次持久化 check point 的时间
  private long mLastCheckTime = 0;
  public void initialize(int shardId)
  {
      mShardId = shardId;
  }
  // 消费数据的主逻辑，这里面的所有异常都需要捕获，不能抛出去。
  public String process(List<LogGroupData> logGroups,
          ILogHubCheckPointTracker checkPointTracker)
  {
      // 这里简单的将获取到的数据打印出来
      for(LogGroupData logGroup: logGroups){
          FastLogGroup flg = logGroup.GetFastLogGroup();
          System.out.println(String.format("\tcategory\t:\t%s\n\
tsource\t:\t%s\n\ttopic\t:\t%s\n\tmachineUUID\t:\t%s",
                  flg.getCategory(), flg.getSource(), flg.getTopic(),
flg.getMachineUUID()));
          System.out.println("Tags");
          for (int tagIdx = 0; tagIdx < flg.getLogTagsCount(); ++
tagIdx) {
              FastLogTag logtag = flg.getLogTags(tagIdx);
              System.out.println(String.format("\t%s\t:\t%s", logtag.
getKey(), logtag.getValue()));
          }
          for (int lIdx = 0; lIdx < flg.getLogsCount(); ++lIdx) {
              FastLog log = flg.getLogs(lIdx);
              System.out.println("--------\nLog: " + lIdx + ", time: "
 + log.getTime() + ", GetContentCount: " + log.getContentsCount());
              for (int cIdx = 0; cIdx < log.getContentsCount(); ++cIdx
) {
                  FastLogContent content = log.getContents(cIdx);
                  System.out.println(content.getKey() + "\t:\t" +
content.getValue());
              }
          }
      }
      long curTime = System.currentTimeMillis();
      // 每隔 30 秒，写一次 check point 到服务端，如果 30 秒内，worker crash
，
```

```
        // 新启动的 worker 会从上一个 checkpoint 其消费数据，有可能有少量的重复
数据
        if (curTime - mLastCheckTime >  30 * 1000)
        {
            try
            {
                //参数true表示立即将checkpoint更新到服务端，为false会将
checkpoint缓存在本地，后台默认隔60s会将checkpoint刷新到服务端。
                checkPointTracker.saveCheckPoint(true);
            }
            catch (LogHubCheckPointException e)
            {
                e.printStackTrace();
            }
            mLastCheckTime = curTime;
        }
        return null;
    }
    // 当 worker 退出的时候，会调用该函数，用户可以在此处做些清理工作。
    public void shutdown(ILogHubCheckPointTracker checkPointTracker)
    {
        //将消费断点保存到服务端。
        try {
            checkPointTracker.saveCheckPoint(true);
        } catch (LogHubCheckPointException e) {
            e.printStackTrace();
        }
    }
}
class SampleLogHubProcessorFactory implements ILogHubProcessorFactory
{
    public ILogHubProcessor generatorProcessor()
    {
        // 生成一个消费实例
        return new SampleLogHubProcessor();
    }
}
```

运行上面的代码，就可以将一个Logstore下面的所有数据打印出来，如果需要多个消费者共同消费一个Logstore，可以按程序注释中说的，修改程序，用同样的消费组名称加不同的消费者名称，启动另外的消费进程。

限制与异常诊断

每个Logstore创建消费组个数的上限为10。超出时将报错ConsumerGroupQuotaExceed。

建议为消费者程序配置log4j，可以帮助您将消费组内部遇到的错误信息抛出来，方便定位异常。放置log4j.properties文件到resources目录下执行程序可以看到类似如下异常：

```
[WARN ] 2018-03-14 12:01:52,747 method:com.aliyun.openservices.loghub.
client.LogHubConsumer.sampleLogError(LogHubConsumer.java:159)
```

```
com.aliyun.openservices.log.exception.LogException: Invalid loggroup
count, (0,1000]
```

一个简单的log4j.properties配置供您参考：

```
log4j.rootLogger = info,stdout
log4j.appender.stdout = org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target = System.out
log4j.appender.stdout.layout = org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern = [%-5p] %d{yyyy-MM-dd
HH:mm:ss,SSS} method:%l%n%m%n
```

状态与报警

1. 在控制台查看消费组状态
2. 通过云监控查看消费组延迟#并配置报警

高级定制

对于普通用户，使用上面的程序就可以消费数据，下面要讨论的内容是一些高级主题。

- 希望消费某个时间开始的数据

  上面代码中的LoghubConfig有两个构造函数：

```
// consumerStartTimeInSeconds参数表示1970之后的秒数，含义是消费这个时间点之
后的数据。
public LogHubConfig(String consumerGroupName,
                    String consumerName,
                    String loghubEndPoint,
                    String project, String logStore,
                    String accessId, String accessKey,
                    int consumerStartTimeInSeconds);
// position是个枚举变量，LogHubConfig.ConsumePosition.BEGIN_CURSOR表示
从最老的数据开始消费，LogHubConfig.ConsumePosition.END_CURSOR表示从最新的
数据开始消费。
public LogHubConfig(String consumerGroupName,
                    String consumerName,
                    String loghubEndPoint,
                    String project, String logStore,
                    String accessId, String accessKey,
                    ConsumePosition position);
```

  可以按照消费需求，使用不同的构造方法，但是注意，如果服务端保存有checkpoint，那么开始消费位置以服务端保存的checkpoint为准。

- 用户使用RAM子账号进行访问

  子用户需要设置消费组相关的RAM权限，设置方法参考*RAM*的文档，需要设置的权限如下：

Action	Resource
log:GetCursorOrData	acs:log:**${regionName}**:**${projectOwnerAliUid}**:project/**${projectName}**/logstore/**${logstoreName}**
log:CreateConsumerGroup	acs:log:**${regionName}**:**${projectOwnerAliUid}**:project/**${projectName}**/logstore/**${logstoreName}**/consumergroup/*
log:ListConsumerGroup	acs:log:**${regionName}**:**${projectOwnerAliUid}**:project/**${projectName}**/logstore/**${logstoreName}**/consumergroup/*
log:ConsumerGroupUpdateCheckPoint	acs:log:**${regionName}**:**${projectOwnerAliUid}**:project/**${projectName}**/logstore/**${logstoreName}**/consumergroup/**${consumerGroupName}**
log:ConsumerGroupHeartBeat	acs:log:**${regionName}**:**${projectOwnerAliUid}**:project/**${projectName}**/logstore/**${logstoreName}**/consumergroup/**${consumerGroupName}**
log:UpdateConsumerGroup	acs:log:**${regionName}**:**${projectOwnerAliUid}**:project/**${projectName}**/logstore/**${logstoreName}**/consumergroup/**${consumerGroupName}**
log:GetConsumerGroupCheckPoint	acs:log:**${regionName}**:**${projectOwnerAliUid}**:project/**${projectName}**/logstore/**${logstoreName}**/consumergroup/**${consumerGroupName}**

- 重置checkpoint

  在一些场景中（补数据、重复计算），我们需要将某个ConsumerGroup点位设置为某一个时间点，使得当前消费组能够从新位置开始消费，有两种方法：

  1. 删除消费组。
     - 停掉消费程序，并在控制台删除消费组。
     - 修改代码，根据上面讨论的使用指定时间点消费，重新启动程序。
  2. 通过SDK将当前消费组重置到某一个时间点。
     - 停掉消费程序。

- 使用sdk修改位点，重新启动消费程序。

```
Client client = new Client(host, accessId, accessKey);
long time_stamp = Timestamp.valueOf("2017-11-15 00:00:00").getTime
() / 1000;
ListShardResponse shard_res = client.ListShard(new ListShardRequest(
project, logStore));
ArrayList<Shard> all_shards = shard_res.GetShards();
for (Shard shard: all_shards)
{
  shardId = shard.GetShardId();
  long cursor_time = time_stamp;
  String cursor = client.GetCursor(project, logStore, shardId,
cursor_time).GetCursor();
  client.UpdateCheckPoint(project, logStore, consumerGroup, shardId
, cursor);
}
```

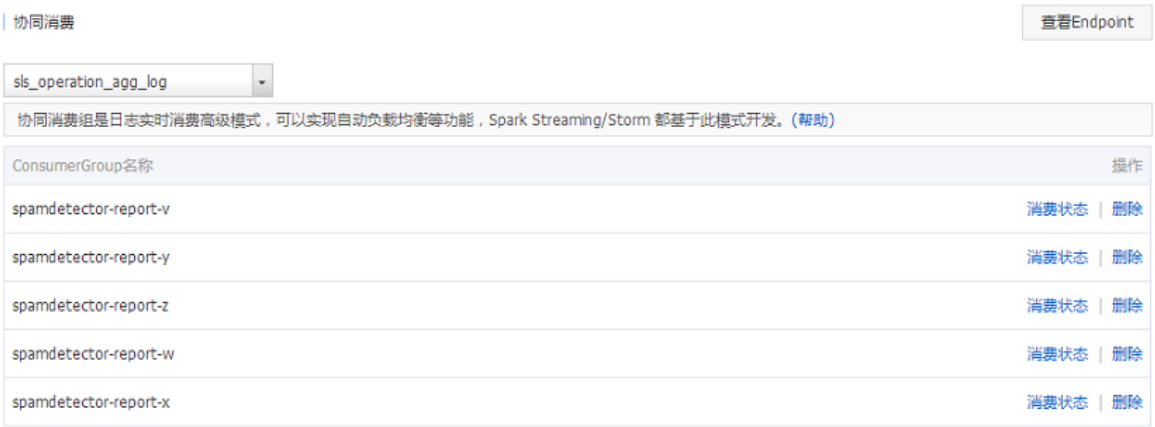
10.3.2 消费组状态

协同消费组#ConsumerGroup# 是实时消费数据高级模式，能够提供多个消费实例对日志库消费自动负载均衡。Spark Streaming、Storm 都以 ConsumerGroup 作为基础模式。

通过控制台查看消费进度

1. 登录日志服务控制台。
2. 选择所需的项目，单击项目名称。
3. 单击左侧导航栏中的 **LogHub - 实时消费 > 协同消费**。
4. 在 **协同消费** 功能页面，选择日志库（Logstore）后即可查看目前是否启用协同消费功能。

   图 10-2: 协同消费



5. 选择指定的 ConsumerGroup 之后，单击 **消费状态**，即可查看当前每个 shard 消费数据的进度。

图 **10-3:** 消费状态



如上图所示，页面上展示该日志库包含 4 个 Shard，对应 4 个消费者，其中每个消费者最近消费的数据时间如第二列显示。通过消费数据时间可以判断出目前数据处理是否能满足数据产生速度，如果已经严重落后于当前时间（即数据消费速率小于数据产生速率），可以考虑增加消费者数目。

通过 API/SDK 查看消费进度

以 Java SDK 作为例子，演示如何通过 API 获得消费状态。

```
package test;
import java.util.ArrayList;
import com.aliyun.openservices.log.Client;
import com.aliyun.openservices.log.common.Consts.CursorMode;
import com.aliyun.openservices.log.common.ConsumerGroup;
import com.aliyun.openservices.log.common.ConsumerGroupShardCheckPoint
;
import com.aliyun.openservices.log.exception.LogException;
public class ConsumerGroupTest {
    static String endpoint = "";
    static String project = "";
    static String logstore = "";
    static String accesskeyId = "";
    static String accesskey = "";
    public static void main(String[] args) throws LogException {
        Client client = new Client(endpoint, accesskeyId, accesskey);
        //获取这个 logstore 下的所有 consumer group，可能不存在，此时
consumerGroups 的长度是 0
        ArrayList<ConsumerGroup>  consumerGroups;
        try{
            consumerGroups = client.ListConsumerGroup(project,
logstore).GetConsumerGroups();
        }
        catch(LogException e){
            if(e.GetErrorCode() == "LogStoreNotExist")
                System.out.println("this logstore does not have any
consumer group");
            else{
                //internal server error branch
```

```
            }
            return;
        }
        for(ConsumerGroup c: consumerGroups){
            //打印 consumer group 的属性，包括名称、心跳超时时间、是否按序消费
            System.out.println("名称: " + c.getConsumerGroupName());
            System.out.println("心跳超时时间: " + c.getTimeout());
            System.out.println("按序消费: " + c.isInOrder());
            for(ConsumerGroupShardCheckPoint cp: client.GetCheckPoint(
project, logstore, c.getConsumerGroupName()).GetCheckPoints()){
                System.out.println("shard: " + cp.getShard());
                //请格式化下，这个时间返回精确到毫秒的时间，长整型
                System.out.println("最后一次消费数据的时间: " + cp.
getUpdateTime());
                System.out.println("消费者名称: " + cp.getConsumer());
                String consumerPrg = "";
                if(cp.getCheckPoint().isEmpty())
                    consumerPrg = "尚未开始消费";
                else{
                    //unix 时间戳，单位是秒，输出的时候请注意格式化
                    try{
                        int prg = client.GetPrevCursorTime(project,
logstore, cp.getShard(), cp.getCheckPoint()).GetCursorTime();
                        consumerPrg = "" + prg;
                    }
                    catch(LogException e){
                        if(e.GetErrorCode() == "InvalidCursor")
                            consumerPrg = "非法，前一次消费时刻已经超出了
logstore中数据的生命周期";
                        else{
                            //internal server error
                            throw e;
                        }
                    }
                }
                System.out.println("消费进度: " + consumerPrg);
                String endCursor = client.GetCursor(project, logstore
, cp.getShard(), CursorMode.END).GetCursor();
                int endPrg = 0;
                try{
                    endPrg = client.GetPrevCursorTime(project,
logstore, cp.getShard(), endCursor).GetCursorTime();
                }
                catch(LogException e){
                    //do nothing
                }
                //unix 时间戳，单位是秒，输出的时候请注意格式化
                System.out.println("最后一条数据到达时刻: " + endPrg);
            }
        }
    }
```

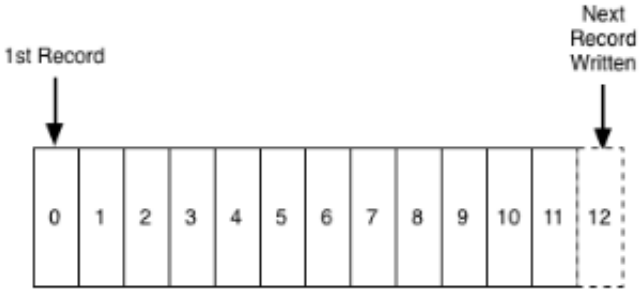
```
}
```

10.3.3 消费组监控与报警

ConsumerGroup 是一个消费者组，包含多个Consumer，每个Consumer消费Logstore中的一部分Shard。

Shard的数据模型可以简单理解成一个队列，新写入的数据被加到队尾，队列中的每条数据都会对应一个数据写入时间，下图是Shard的数据模型。

图 **10-4: Shard**数据模型



协同消费延迟报警中的基本概念：

- **消费过程**：消费者从队头开始顺序读取数据的过程。
- **消费进度**：消费者当前读取的数据对应的写入时间。
- **消费落后时长**：当前消费进度和队列中最新的数据写入时间的差值，单位为秒。

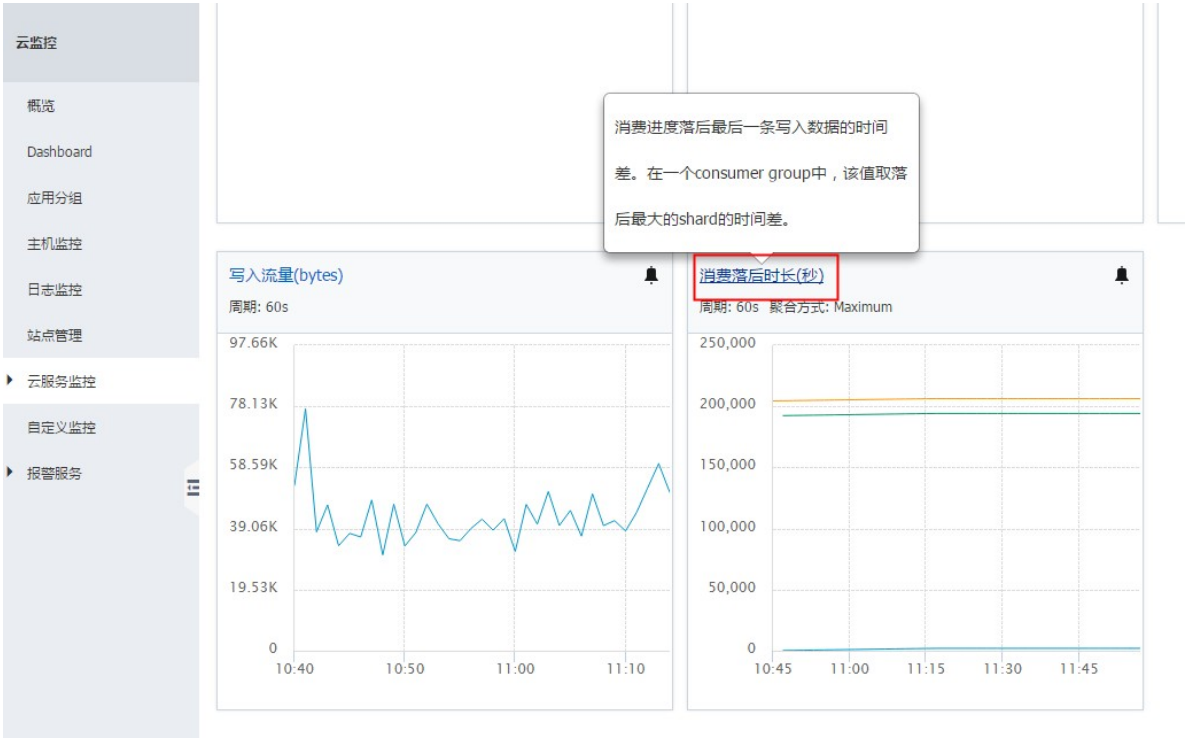
ConsumerGroup的消费落后时长取其包含的所有Shard的消费落后时长的最大值，当超过用户预设阈值时，就认为消费落后太多，需要报警。

操作步骤

操作步骤

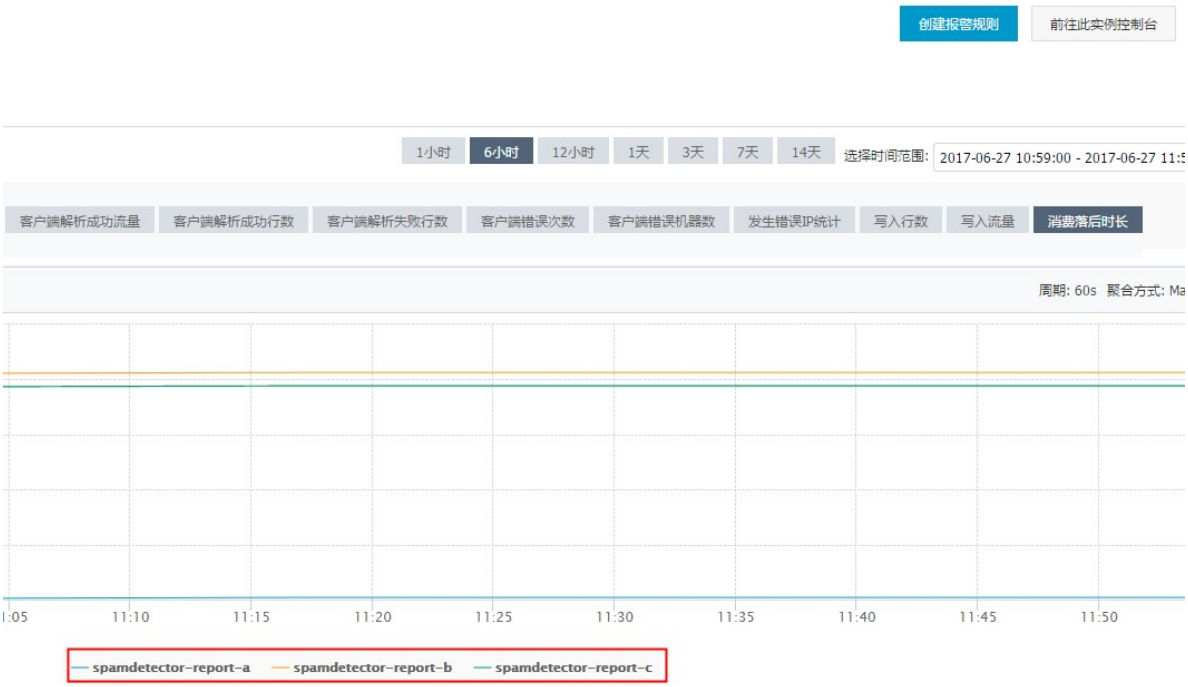
1. 登录日志服务控制台，单击需要监控的 Logstore 的监控图标。
2. 找到消费落后时长图表，单击进入云监控控制台。

图 **10-5:** 进入云监控控制台



3. 该图展示了 Logstore 下所有 ConsumerGroup 的消费落后时长，单位为秒。框中图例是所有的 ConsumerGroup，单击右上角 创建报警规则 进入规则创建页面。

图 **10-6:** 查看消费落后时长



4. 创建针对 ConsumerGroup spamdetector-report-c 的报警规则，5分钟 内只要有一次延迟大于等于 600 秒就会报警。设置生效时间和报警通知联系人，保存规则。

图 10-7: 设置报警规则



上面的操作完成后便成功创建了报警规则。有关报警规则配置的任何问题，请提工单到云监控。

10.4 函数计算消费

10.4.1 开发指南

日志服务自定义ETL功能的数据消费端运行在阿里云函数计算服务上，根据不同的ETL用途，您可以选择使用日志服务提供的函数模板，或是自定义实现个性化函数。

本文介绍如何实现一个自定义日志服务ETL函数。

函数Event

函数Event即运行函数的输入参数，格式为一个JSON Object序列化后字符串。

字段说明

- **jobName字段**

  日志服务ETL Job名字，函数计算服务上的日志服务触发器对应一个日志服务的ETL Job。

- **taskId字段**

  对于一个ETL Job，taskId是某一次确定性的函数调用的标识。

- **cursorTime字段**

  本次函数调用包括的数据中，最后一条日志到达日志服务服务端的unix_timestamp。

- **source字段**

  该字段由日志服务生成，日志服务根据ETL Job定义的任务间隔定时触发函数执行，source字段是函数Event中很重要的一部分，定义了本次函数调用应该消费哪些数据。

  该数据源范围由如下字段（相关字段定义请参考日志服务概念介绍）组成：

字段	说明
endpoint	Project所属日志服务Region的服务入口
projectName	Project名称
logstoreName	Logstore名称
shardId	Logstore下的某一个确定Shard
beginCursor	需要从Shard的什么位置开始消费数据
endCursor	需要消费Shard数据到什么位置

   说明：

  Shard对应的[beginCursor, endCursor)是一个左闭右开区间。

- **parameter字段**

  JSON Object类型，由用户在创建ETL Job（函数计算的LOG触发器）时设定。自定义函数运行时解析该字段，可以获取到函数所需要的运行参数。

  在函数计算控制台创建日志服务触发器时，通过填写**函数配置**进行设置，如下图：

图 10-8: 函数配置



函数Event示例

```
{
    "source": {
        "endpoint": "http://cn-shanghai-intranet.log.aliyuncs.com",
        "projectName": "fc-1584293594287572",
        "logstoreName": "demo",
        "shardId": 0,
        "beginCursor": "MTUwNTM5MDI3NTY1ODcwNzU2Ng==",
        "endCursor": "MTUwNTM5MDI3NTY1ODcwNzU2OA=="
    },
    "parameter": {
        ...
    },
    "jobName": "fedad35f51a2a97b466da57fd71f315f539d2234",
```

```
    "taskId": "9bc06c96-e364-4f41-85eb-b6e579214ae4",
    "cursorTime": 1511429883
}
```

在函数调试时候，你也可以根据*GetCursor*接口获取cursor并按上述格式自行组装一个函数Event用于测试。

函数开发

您可以使用Java/Python/Node.js等多种语言实现函数功能，日志服务提供了相应runtime的各种语言*SDK*以便您在函数中进行集成。

本节以Java8 runtime为例，介绍如何开发日志服务ETL函数。由于涉及Java8函数编程细节，请先阅读函数计算服务*Java*编程指南。

Java函数模板

目前，日志服务提供了基于Java8执行环境的自定义*ETL*函数模板，您可以在这基础上完成自定义需求的实现。

模板已实现以下功能：

- 函数Event中source、taskId、jobName字段的解析。
- 根据source中定义的数据源，通过日志服务*Java SDK*拉取数据，并对每一批数据调用processData接口进行处理。

在模板中，您还需要实现以下功能：

- 函数Event中parameter字段的解析，`UserDefinedFunctionParameter.java`实现。
- 函数内针对数据的自定义业务逻辑，`UserDefinedFunction.java`的processData接口实现。
- 为您的函数取一个可以恰当描述功能的名字，替换`UserDefinedFunction`。

processData方法实现

您需要在processData内完成对一批数据的消费、加工、投递，视具体需求而定。

可以参考*LogstoreReplication*，它实现了将数据从一个Logstore中读出后写到另一个日志服务Logstore。

note



说明：

1. processData处理数据成功请返回true，处理数据遇到异常无法重试成功的请返回false，但此时函数还会继续运行下去，且日志服务判定这是一次成功的ETL任务，只是忽略了一些处理异常的数据。
2. 如果遇到致命错误或业务逻辑上认为有异常需要提前终止函数执行，请通过throw Exception方式跳出函数运行，日志服务可以据此判断函数运行异常，并会按照ETL Job设置的规则重新调用函数执行。

注意事项

- 如果Shard流量较大，请注意为函数配置足够的内存规格，以避免函数OOM导致异常终止。
- 如果函数内执行耗时操作，或者Shard流量较大，请注意设置较短的函数触发间隔和较长的函数运行超时时间。
- 请为函数服务配置足够的权限，例如在函数内写OSS就需要配置OSS写权限。

ETL日志

- **ETL调度日志**

  调度日志只记录ETL任务开始的时间、结束时间，任务是否成功以及成功返回的信息。如果ETL任务出错了，不仅要生成ETL出错日志，而且要向系统管理员发送报警邮件或短信。在创建触发器时设置触发器日志Logstore，并为该Logstore开通索引查询功能。

  对于函数执行的统计，可以通过函数写出返回，例如Java8函数的outputStream。日志服务默认提供的模板会写出一个JSON Object序列化后的字符串，该字符串将一并记录在ETL任务调度日志中，方便您进行统计、查询。

- **ETL过程日志**

  这一部分日志是在ETL执行过程中每执行一步的记录关键点和错误，包括某一步骤的开始、结束时间、初始化动作完成情况，模块出错信息等。ETL过程日志的意义是随时可以感知ETL运行情况，如果发生错误，可以及时通过过程日志查找原因。

  您可以通过context.getLogger()记录过程日志到日志服务指定的Project、Logstore，建议为该Logstore开启索引查询功能。

10.4.2 配置函数计算消费日志

依托于阿里云函数计算服务，日志服务提供流式的全托管数据加工服务。

您可以配置一个ETL Job，日志服务将定时获取数据更新并触发函数执行：增量消费日志服务Logstore的数据，在函数里完成自定义加工任务。用于数据加工的函数可以是日志服务提供的模板或者用户自定义函数。

适用场景

数据清洗、加工场景

通过日志服务，快速完成日志采集、加工、查询、分析。

图 **10-9:** 数据清洗、加工场景



数据投递场景

为数据的目的端落地提供支撑，构建云上大数据产品间的数据管道。

图 **10-10:** 数据投递场景



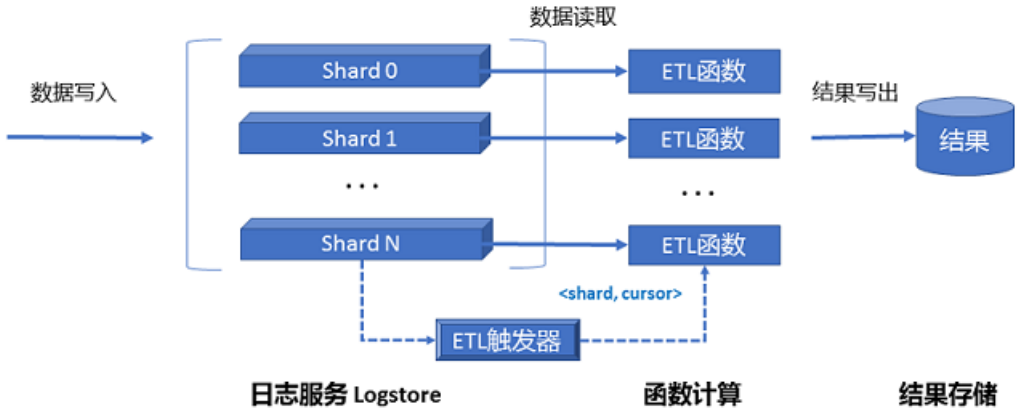
工作原理

触发机制

日志服务的ETL Job对应于函数计算的一个触发器，创建ETL Job后，日志服务将根据该Job配置启动定时器，定时器轮询Logstore的shard信息，当发现有新日志写入时，生成的< shard_id, begin_cursor, end_cursor >三元组信息作为函数event，触发函数执行。

日志服务的ETL任务触发机制是时间触发，例如：ETL Job触发间隔为60秒，Logstore的Shard 0一直有数据写入，那么对于Shard 0每1分钟就会触发一次函数执行（如果Shard没有新的数据写入则不会触发函数执行），函数执行的输入为最近60秒的cursor区间。在函数内，可以根据cursor读取Shard 0数据进行处理。

图 **10-11:** 触发机制



ETL函数

您可以选择使用函数模板或者自定义函数，开始前建议先了解函数计算服务的基本概念。

- 日志服务维护的函数模板

  函数模板在GitHub上维护，请单击*aliyun-log-fc-functions*访问。

- 用户自定义函数

  由您自行实现，函数配置的格式与函数的具体实现有关，请参考*ETL*函数开发指南。

使用指南

步骤1 授权与准备资源

1. 在快捷授权页面单击**同意授权**，为日志服务触发函数执行授权。
2. 创建日志服务Project和函数过程日志Logstore。

   如您之前从未创建过Project和Logstore，请参考准备流程创建。

   说明：
   日志服务Project和函数计算Service所属区域必须相同。

步骤2 创建服务

1. 在函数计算控制台右上角单击**新建服务**。
2. 填写**服务名称**和**功能描述**，并展开**高级配置**。

配置项	说明
服务名称	为您新建的函数计算服务命名。命名规则： • 由英文字符（a-z）或（A-Z），数字（0-9），下划线（_）和中划线（-）组成 • 首字母必须为英文字母（a-z）、（A-Z）或下划线（_） • 大小写敏感，且长度为1-128字符
功能描述	新建服务的描述内容。
日志项目	日志服务Project名称。必须与您新建函数计算服务的项目位于同一Region。
日志仓库	日志服务Logstore名称。必须与您新建函数计算服务的项目位于同一Region。
角色创建方式	创建服务角色，根据选择的系统模板创建对应的权限。授予函数计算推送日志到指定的仓库。您可以新建角色或是选择已有角色。如您选择使用已有角色，需要选定一个已经存在的角色。
系统模板授权	选择系统授权策略。日志服务支持`AliyunLogFullAccess`和`AliyunLogReadOnlyAccess`两个系统授权策略。

图 10-12: 创建服务



确定系统授权策略之后，单击**点击授权**。此时会跳出**角色快捷创建**弹窗，您需要在弹窗中确认**角色信息**和**权限信息**，包括**策略名称**、**策略描述**和**策略详情**。如果是新建策略，您还需要确认**角色名称**和**角色描述**。在**策略详情**中，您可以细化授权策略，为该角色自定义一个合适的授权策略。

服务授权成功后，单击**确定**，进入**服务概览**界面。

步骤3 创建函数和触发器

1. 在**服务概览**界面，单击**新建函数**。

   选择**函数模板**。

   在函数创建向导中，您可以选择与您业务模型相似的业务模板进行修改并创建函数，也可以选择空白函数模板自定义函数。

- 日志服务模板。`logstore_replication`和`oss-shipper-csv`是日志服务提供的业务模板，您可以基于此模板创建函数及触发器。
- 空白模板。空白函数模板会创建一个空白函数，通过引导页面进行触发器配置、函数参数配置和代码开发，完成函数的创建。

2. 填写**触发器配置**，并单击**下一步**。

如果您选择了日志服务提供的模板，您可以直接配置触发器；如果您选择了空白模板，请先选择触发器类型，再配置触发器。

请填写触发器名称、Project名称、Logstore名称必选项，配置触发器。一个函数计算的LOG类型触发器对应一个日志服务的ETL Job。

配置项	说明	取值
触发器名称	为您新建的触发器命名。	触发器的名称只能包含字母，数字、下划线和中划线，不能以数字、中划线开头，且长度范围为1~256字节。
Project名称	日志服务Project名称。	必须是已经存在的Project。该Project必须与您的服务处于同一地域。
Logstore名称	日志服务Logstore名称（数据源）。本触发器会定时从该Logstore订阅数据到函数计算进行自定义加工，该参数在ETL Job创建后不允许修改。	必须是已存在的Logstore，且属于**Project名称**中已配置的Project。
触发器日志	日志服务会定时触发函数计算的的函数执行，在触发过程中发生的异常、函数执行统计信息会记录到该 Logstore， 您可以为这个Logstore创建索引以备查看。	必须是已存在的Logstore，且属于**Project名称**中已配置的Project。
触发间隔	日志服务触发函数运行的间隔，定义日志服务触发函数执行的间隔。例如每60秒将logstore的每个shard最近60秒数据位置读出作为函数event	取值范围为3~600，单位为秒。

配置项	说明	取值
	调用函数执行，在函数内有用户逻辑读取shard数据做计算。如果logstore的shard流量较大（超过1MB/s或者更高），建议缩短函数的触发间隔，使得每次函数运行所处理的数据量是合理大小。	
重试次数	日志服务根据触发间隔每次触发函数执行时，如果遇到错误（例如权限不足、网络失败、函数执行异常返回等），该参数定义本次触发所允许的最大重试次数。对于本次触发，如果超过最大重试次数仍无法成功的，需要等到下一次触发间隔到来时，由日志服务再次触发函数执行。重试对业务造成的影响，因具体的函数代码实现逻辑而异。	取值范围为0~100，单位为次。
函数配置	日志服务将该配置内容作为函数event一部分传入函数，如何使用该函数由函数自定义逻辑决定。每一种函数实现所要求的函数配置可能是不同的，绝大部分默认提供的函数模板也需要参考说明填写您的参数。默认不传入任何参数时请填写：`{}`。	该配置内容必须是JSON Object格式字符串。

图 10-13: 触发器配置



说明：

您已经拥有了让日志服务调用您的函数和读写Logstore的的权限。

3. 填写基础管理配置。

   填写函数名称、函数入口等基础管理配置信息，并单击下一步。

4. 填写权限配置。

   确认模版授权和触发器角色授权，并单击下一步。

5. 核对您的函数信息和触发器信息，并单击完成。

查看触发器日志

登录日志服务控制台，为Job配置的触发器日志Logstore创建索引，查看任务执行统计结果。

查看函数运行日志

登录日志服务控制台，查看函数执行过程的详细信息，详细信息请参考日志记录。

常见问题

创建触发器后未触发函数执行

1. 确认已经使用快捷授权为日志服务触发函数执行授权。
2. 确认Job配置的Logstore是否有数据增量修改，当Shard数据有变化时会触发函数执行。
3. 登录日志服务控制台查看触发器日志、函数运行日志查看是否有异常。

10.5 Flink 消费

Flink log connector是阿里云日志服务提供的，用于对接Flink的工具，包括两部分，消费者（Consumer）和生产者（Producer）。

消费者用于从日志服务中读取数据，支持exactly once语义，支持Shard负载均衡。

生产者用于将数据写入日志服务，使用connector时，需要在项目中添加maven依赖：

```
<dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-streaming-java_2.11</artifactId>
            <version>1.3.2</version>
</dependency>
<dependency>
            <groupId>com.aliyun.openservices</groupId>
            <artifactId>flink-log-connector</artifactId>
            <version>0.1.7</version>
</dependency>
<dependency>
            <groupId>com.google.protobuf</groupId>
            <artifactId>protobuf-java</artifactId>
            <version>2.5.0</version>
</dependency>
 <dependency>
            <groupId>com.aliyun.openservices</groupId>
            <artifactId>aliyun-log</artifactId>
            <version>0.6.19</version>
 </dependency>
<dependency>
            <groupId>com.aliyun.openservices</groupId>
            <artifactId>log-loghub-producer</artifactId>
            <version>0.1.8</version>
</dependency>
```

前提条件

1. 已启用Access Key，并创建了Project和Logstore。详细步骤请参考准备流程。

2. 若您选择使用子账号操作日志服务，请确认已正确设置了Logstore的RAM授权策略。详细内容请参考授权*RAM* 用户。

Log Consumer

在Connector中， 类FlinkLogConsumer提供了订阅日志服务中某一个Logstore的能力，实现了exactly once语义，在使用时，用户无需关心Logstore中shard数量的变化，consumer会自动感知。

Flink中每一个子任务负责消费Logstore中部分shard，如果Logstore中shard发生split或者merge，子任务消费的shard也会随之改变。

关联 **API**

Flink log consumer 会用到的阿里云日志服务接口如下：

- GetCursorOrData

  用于从shard中拉数据， 注意频繁的调用该接口可能会导致数据超过日志服务的shard quota，可以通过ConfigConstants.LOG_FETCH_DATA_INTERVAL_MILLIS和ConfigConstants.LOG_MAX_NUMBER_PER_FETCH控制接口调用的时间间隔和每次调用拉取的日志数量，shard的quota参考文章*shard*简介.

  ```
  configProps.put(ConfigConstants.LOG_FETCH_DATA_INTERVAL_MILLIS， "100");
  configProps.put(ConfigConstants.LOG_MAX_NUMBER_PER_FETCH， "100");
  ```

- ListShards

  用于获取Logstore中所有的shard列表，获取shard状态等.如果您的shard经常发生分裂合并，可以通过调整接口的调用周期来及时发现shard的变化。

  ```
  // 设置每30s调用一次ListShards
  configProps.put(ConfigConstants.LOG_SHARDS_DISCOVERY_INTERVAL_MILLIS， "30000");
  ```

- CreateConsumerGroup

  该接口调用只有当设置消费进度监控时才会发生，功能是创建consumerGroup，用于同步checkpoint。

- ConsumerGroupUpdateCheckPoint

  该接口用户将flink的snapshot同步到日志服务的consumerGroup中。

子用户权限

子用户使用Flink log consumer需要授权如下几个RAM Policy：

接口	资源
log:GetCursorOrData	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}
log:ListShards	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}
log:CreateConsumerGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/*
log:ConsumerGroupUpdateCheckPoint	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/${consumerGroupName}

配置步骤

1 配置启动参数

```
Properties configProps = new Properties();
// 设置访问日志服务的域名
configProps.put(ConfigConstants.LOG_ENDPOINT, "cn-hangzhou.log.aliyuncs.com");
// 设置访问ak
configProps.put(ConfigConstants.LOG_ACCESSSKEYID, "");
configProps.put(ConfigConstants.LOG_ACCESSKEY, "");
// 设置日志服务的project
configProps.put(ConfigConstants.LOG_PROJECT, "ali-cn-hangzhou-sls-admin");
// 设置日志服务的Logstore
configProps.put(ConfigConstants.LOG_LOGSTORE, "sls_consumergroup_log");
// 设置消费日志服务起始位置
configProps.put(ConfigConstants.LOG_CONSUMER_BEGIN_POSITION, Consts.LOG_END_CURSOR);
// 设置日志服务的消息反序列化方法
RawLogGroupListDeserializer deserializer = new RawLogGroupListDeserializer();
final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
DataStream<RawLogGroupList> logTestStream = env.addSource(
        new FlinkLogConsumer<RawLogGroupList>(deserializer, configProps));
```

上面是一个简单的消费示例，我们使用java.util.Properties作为配置工具，所有Consumer的配置都可以在ConfigConstants中找到。

 **说明：**

Flink stream的子任务数量和日志服务Logstore中的shard数量是独立的，如果shard数量多于子任务数量，每个子任务不重复的消费多个shard，如果少于子任务数量，那么部分子任务就会空闲，等到新的shard产生。

2 设置消费起始位置

Flink log consumer支持设置shard的消费起始位置，通过设置属性ConfigConstants.LOG_CONSUMER_BEGIN_POSITION，就可以定制消费从shard的头尾或者某个特定时间开始消费，另外，connector也支持从某个具体的ConsumerGroup中恢复消费。具体取值如下：

- Consts.LOG_BEGIN_CURSOR： 表示从shard的头开始消费，也就是从shard中最旧的数据开始消费。
- Consts.LOG_END_CURSOR： 表示从shard的尾开始，也就是从shard中最新的数据开始消费。
- Consts.LOG_FROM_CHECKPOINT：表示从某个特定的ConsumerGroup中保存的Checkpoint开始消费，通过ConfigConstants.LOG_CONSUMERGROUP指定具体的ConsumerGroup。
- UnixTimestamp： 一个整型数值的字符串，用1970-01-01到现在的秒数表示， 含义是消费shard中这个时间点之后的数据。

四种取值举例如下：

```
configProps.put(ConfigConstants.LOG_CONSUMER_BEGIN_POSITION, Consts.
LOG_BEGIN_CURSOR);
configProps.put(ConfigConstants.LOG_CONSUMER_BEGIN_POSITION, Consts.
LOG_END_CURSOR);
configProps.put(ConfigConstants.LOG_CONSUMER_BEGIN_POSITION, "
1512439000");
configProps.put(ConfigConstants.LOG_CONSUMER_BEGIN_POSITION, Consts.
LOG_FROM_CHECKPOINT);
```

 说明：

如果在启动Flink任务时，设置了从Flink自身的StateBackend中恢复，那么connector会忽略上面的配置，使用StateBackend中保存的Checkpoint。

3 设置消费进度监控（可选）

Flink log consumer支持设置消费进度监控，所谓消费进度就是获取每一个shard实时的消费位置，这个位置使用时间戳表示，详细概念可以参考文档消费组状态，消费组监控与报警。

```
configProps.put(ConfigConstants.LOG_CONSUMERGROUP, "your consumer
group name");
```



说明：

以上配置为可选项，如果设置，consumer会首先创建consumerGroup。如果consumerGroup已经存在，则不执行任何操作，consumer中的snapshot会自动同步到日志服务的consumerGroup中，用户可以在日志服务的控制台查看consumer的消费进度。

4 设置容灾和**exactly once**语义支持

当打开Flink的checkpointing功能时，Flink log consumer会周期性的将每个shard的消费进度保存起来，当作业失败时，flink会恢复log consumer，并从保存的最新的checkpoint开始消费。

写checkpoint的周期定义了当发生失败时，最多多少的数据会被回溯，即重新消费，使用代码如下：

```
final StreamExecutionEnvironment env = StreamExecutionEnvironment.
getExecutionEnvironment();
// 开启flink exactly once语义
env.getCheckpointConfig().setCheckpointingMode(CheckpointingMode.
EXACTLY_ONCE);
// 每5s保存一次checkpoint
env.enableCheckpointing(5000);
```

更多Flink checkpoint的细节请参考Flink官方文档*Checkpoints*。

Log Producer

FlinkLogProducer 用于将数据写到阿里云日志服务中。



说明：

Producer只支持Flink at-least-once语义，在发生作业失败的情况下，写入日志服务中的数据有可能会重复，但是不会丢失。

子用户权限

Producer依赖日志服务的API写数据，例如：

- log:PostLogStoreLogs

- log:ListShards

当RAM子用户使用Producer时，需要对以上两个API进行授权：

接口	资源
log:PostLogStoreLogs	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}
log:ListShards	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}

配置步骤

1. 初始化Producer。

   a. 初始化配置参数Properties。

   Producer初始化步骤与Consumer类似。 Producer包含以下参数，一般情况下使用默认值即可，如有需要，可以自定义配置。

```
// 用于发送数据的io线程的数量，默认是8
ConfigConstants.LOG_SENDER_IO_THREAD_COUNT
// 该值定义日志数据被缓存发送的时间，默认是3000
ConfigConstants.LOG_PACKAGE_TIMEOUT_MILLIS
// 缓存发送的包中日志的数量，默认是4096
ConfigConstants.LOG_LOGS_COUNT_PER_PACKAGE
// 缓存发送的包的大小，默认是3Mb
ConfigConstants.LOG_LOGS_BYTES_PER_PACKAGE
// 作业可以使用的内存总的大小，默认是100Mb
ConfigConstants.LOG_MEM_POOL_BYTES
```

   上述参数不是必选参数，用户可以不设置，直接使用默认值。

   b. 重载LogSerializationSchema，定义将数据序列化成RawLogGroup的方法。

   RawLogGroup是log的集合，每个字段的含义可以参考文档数据模型。

   如果用户需要使用日志服务的shardHashKey功能，指定数据写到某一个shard中，可以使用LogPartitioner产生数据的hashKey。

   示例：

```
FlinkLogProducer<String> logProducer = new FlinkLogProducer<String
>(new SimpleLogSerializer(), configProps);
logProducer.setCustomPartitioner(new LogPartitioner<String>() {
      // 生成32位hash值
      public String getHashKey(String element) {
```

```
            try {
                MessageDigest md = MessageDigest.getInstance("MD5");
                md.update(element.getBytes());
                String hash = new BigInteger(1, md.digest()).
toString(16);
                while(hash.length() < 32) hash = "0" + hash;
                return hash;
            } catch (NoSuchAlgorithmException e) {
            }
            return  "00000000000000000000000000000000000000000000
0000000000000000000000";
        }
    });
```



说明：

LogPartitioner为可选项，如您没有配置，数据会随机写入某一个Shard。

2. 执行以下示例语句，将模拟产生的字符串写入日志服务。

```
// 将数据序列化成日志服务的数据格式
class SimpleLogSerializer implements LogSerializationSchema<String>
 {
    public RawLogGroup serialize(String element) {
        RawLogGroup rlg = new RawLogGroup();
        RawLog rl = new RawLog();
        rl.setTime((int)(System.currentTimeMillis() / 1000));
        rl.addContent("message", element);
        rlg.addLog(rl);
        return rlg;
    }
}
public class ProducerSample {
    public static String sEndpoint = "cn-hangzhou.log.aliyuncs.com";
    public static String sAccessKeyId = "";
    public static String sAccessKey = "";
    public static String sProject = "ali-cn-hangzhou-sls-admin";
    public static String sLogstore = "test-flink-producer";
    private static final Logger LOG = LoggerFactory.getLogger(
ConsumerSample.class);
    public static void main(String[] args) throws Exception {
        final ParameterTool params = ParameterTool.fromArgs(args);
        final StreamExecutionEnvironment env = StreamExecutionEnvir
onment.getExecutionEnvironment();
        env.getConfig().setGlobalJobParameters(params);
        env.setParallelism(3);
        DataStream<String> simpleStringStream = env.addSource(new
EventsGenerator());
        Properties configProps = new Properties();
        // 设置访问日志服务的域名
        configProps.put(ConfigConstants.LOG_ENDPOINT, sEndpoint);
        // 设置访问日志服务的ak
        configProps.put(ConfigConstants.LOG_ACCESSSKEYID,
sAccessKeyId);
        configProps.put(ConfigConstants.LOG_ACCESSKEY, sAccessKey);
        // 设置日志写入的日志服务project
        configProps.put(ConfigConstants.LOG_PROJECT, sProject);
        // 设置日志写入的日志服务Logstore
        configProps.put(ConfigConstants.LOG_LOGSTORE, sLogstore);
```

```
        FlinkLogProducer<String> logProducer = new FlinkLogProducer<
String>(new SimpleLogSerializer(), configProps);
        simpleStringStream.addSink(logProducer);
        env.execute("flink log producer");
    }
    // 模拟产生日志
    public static class EventsGenerator implements SourceFunction<
String> {
        private boolean running = true;
        @Override
        public void run(SourceContext<String> ctx) throws Exception
 {
            long seq = 0;
            while (running) {
                Thread.sleep(10);
                ctx.collect((seq++) + "-" + RandomStringUtils.
randomAlphabetic(12));
            }
        }
        @Override
        public void cancel() {
            running = false;
        }
    }
}
```

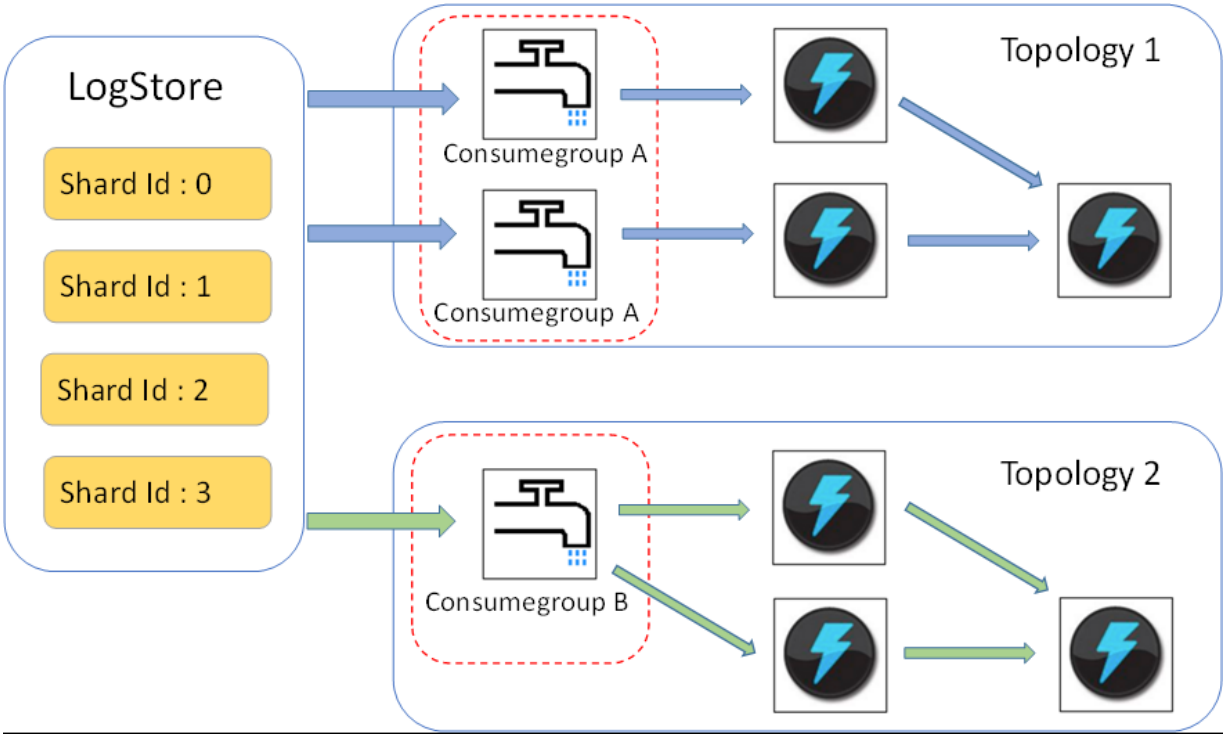
10.6 Storm消费

日志服务的LogHub提供了高效、可靠的日志通道功能，您可以通过Logtail、SDK等多种方式来实时收集日志数据。收集日志之后，可以通过Spark Stream、Storm 等各实时系统来消费写入到LogHub中的数据。

为了降低Storm用户消费LogHub的代价，日志服务提供了LogHub Storm Spout来实时读取LogHub的数据。

基本结构和流程

图 **10-14:** 基本结构和流程



- 上图中红色虚线框中就是LogHub Storm Spout，每个Storm Topology会有一组Spout，同组内的Spout共同负责读取Logstore中全部数据。不同Topology中的Spout相互不干扰。
- 每个Topology需要选择唯一的LogHub Consume Group名字来相互标识，同一 Topology内的Spout通过 *Consumer Library* 来完成负载均衡和自动failover。
- Spout从LogHub中实时读取数据之后，发送至Topology中的Bolt节点，定期保存消费完成位置作为checkpoint到LogHub服务端。

使用限制

- 为了防止滥用，每个Logstore最多支持 5 个Consumer Group，对于不再使用的 Consumer Group，可以使用Java SDK中的DeleteConsumerGroup接口进行删除。
- Spout的个数最好和Shard个数相同，否则可能会导致单个Spout处理数据量过多而处理不过来。
- 如果单个Shard 的数据量太大，超过一个Spout处理极限，则可以使用Shard split接口分裂Shard，来降低每个Shard的数据量。
- 在Loghub Spout中，强制依赖Storm的ACK机制，用于确认Spout将消息正确发送至Bolt，所以在Bolt中一定要调用ACK进行确认。

使用样例

- **Spout** 使用示例（用于构建 **Topology**）

```
     public static void main( String[] args )
    {
        String mode = "Local";  // 使用本地测试模式
           String conumser_group_name = "";   // 每个Topology 需要设定唯一的 consumer group 名字，不能为空，支持 [a-z][0-9] 和 '_'，'-'，长度在 [3-63] 字符，只能以小写字母和数字开头结尾
        String project = "";    // 日志服务的Project
        String logstore = "";   // 日志服务的Logstore
        String endpoint = "";   // 日志服务访问域名
        String access_id = "";  // 用户 ak 信息
        String access_key = "";
        // 构建一个 Loghub Storm Spout 需要使用的配置
        LogHubSpoutConfig config = new LogHubSpoutConfig(conumser_group_name,
                endpoint, project, logstore, access_id,
                access_key, LogHubCursorPosition.END_CURSOR);
        TopologyBuilder builder = new TopologyBuilder();
        // 构建 loghub storm spout
        LogHubSpout spout = new LogHubSpout(config);
        // 在实际场景中，Spout的个数可以和Logstore Shard 个数相同
        builder.setSpout("spout", spout, 1);
        builder.setBolt("exclaim", new SampleBolt()).shuffleGrouping("spout");
        Config conf = new Config();
        conf.setDebug(false);
        conf.setMaxSpoutPending(1);
        // 如果使用Kryo进行数据的序列化和反序列化，则需要显示设置 LogGroupData 的序列化方法 LogGroupDataSerializSerializer
        Config.registerSerialization(conf, LogGroupData.class, LogGroupDataSerializSerializer.class);
        if (mode.equals("Local")) {
            logger.info("Local mode...");
            LocalCluster cluster  = new LocalCluster();
            cluster.submitTopology("test-jstorm-spout", conf, builder.createTopology());
            try {
                Thread.sleep(6000 * 1000);    //waiting for several minutes
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            cluster.killTopology("test-jstorm-spout");
            cluster.shutdown();
        } else if (mode.equals("Remote")) {
            logger.info("Remote mode...");
            conf.setNumWorkers(2);
            try {
                StormSubmitter.submitTopology("stt-jstorm-spout-4", conf, builder.createTopology());
            } catch (AlreadyAliveException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InvalidTopologyException e) {
```

```
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        } else {
            logger.error("invalid mode: " + mode);
        }
    }
}
```

- 消费数据的 **bolt** 代码样例，只打印每条日志的内容

```
public class SampleBolt extends BaseRichBolt {
    private static final long serialVersionUID = 4752656887
774402264L;
    private static final Logger logger = Logger.getLogger(BaseBasicB
olt.class);
    private OutputCollector mCollector;
    @Override
    public void prepare(@SuppressWarnings("rawtypes") Map stormConf
, TopologyContext context,
            OutputCollector collector) {
        mCollector = collector;
    }
    @Override
    public void execute(Tuple tuple) {
        String shardId = (String) tuple
                .getValueByField(LogHubSpout.FIELD_SHARD_ID);
        @SuppressWarnings("unchecked")
        List<LogGroupData> logGroupDatas = (ArrayList<LogGroupData
>) tuple.getValueByField(LogHubSpout.FIELD_LOGGROUPS);
        for (LogGroupData groupData : logGroupDatas) {
            // 每个 logGroup 由一条或多条日志组成
            LogGroup logGroup = groupData.GetLogGroup();
            for (Log log : logGroup.getLogsList()) {
                StringBuilder sb = new StringBuilder();
                // 每条日志，有一个时间字段，以及多个 Key:Value 对，
                int log_time = log.getTime();
                sb.append("LogTime:").append(log_time);
                for (Content content : log.getContentsList()) {
                    sb.append("\t").append(content.getKey()).append
(":")
                            .append(content.getValue());
                }
                logger.info(sb.toString());
            }
        }
        // 在 loghub spout 中，强制依赖 storm 的 ack 机制，用于确认 spout
 将消息正确
        // 发送至 bolt，所以在 bolt 中一定要调用 ack
        mCollector.ack(tuple);
    }
    @Override
    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        //do nothing
    }
```

```
}
```

Maven

storm 1.0 之前版本（如 0.9.6），请使用：

```
<dependency>
  <groupId>com.aliyun.openservices</groupId>
  <artifactId>loghub-storm-spout</artifactId>
  <version>0.6.5</version>
</dependency>
```

storm 1.0 版本及以后，请使用：

```
<dependency>
  <groupId>com.aliyun.openservices</groupId>
  <artifactId>loghub-storm-1.0-spout</artifactId>
  <version>0.1.2</version>
</dependency>
```

10.7 Spark Streaming消费

E-MapReduce 实现了一套通用的Spark Streaming实时消费LogHub的接口，SDK请转至*GitHub*下载。

10.8 StreamCompute消费

StreamCompute 在创建Loghub类型数据源后，可以直接消费Loghub中数据，配置如下：

```
CREATE STREAM TABLE source_test_galaxy ( $schema ) WITH ( type='loghub
',
endpoint=$endpoint, accessId=$loghub_access_id, accessKey=$loghub_acc
ess_key, projectName=$project, logstore=$logstore );
```

表 **10-1:** 参数列表

参数	参数说明
`$schema`	将日志中哪些Key映射成StreamCompute表中的Column，如：`name STRING, age STRING, id STRING`。
`$endpoint`	数据访问接入点，各Region接入点请查看服务入口。
`$loghub_access_id`	有读权限账号（或子账号）对应AccessId。
`$loghub_access_key`	有读权限账号（或子账号）对应AccessKey。
`$project`	数据所在Project。

参数	参数说明
$logstore	数据所在Logstore。

示例：

```
CREATE STREAM TABLE source_test_galaxy ( name STRING, age STRING, id
 STRING ) WITH ( type='loghub', endpoint='http://cn-hangzhou-intranet
.log.aliyuncs.com', accessId='mock_access_id', accessKey='mock_acces
s_key', projectName='ali-cloud-streamtest', logstore='stream-test' );
```

10.9 CloudMonitor消费

云监控*(Cloud Monitor)*可以直接消费LogHub下Logstore数据，提供监控功能。

主要功能：

- 对日志中的关键字报警。
- 统计单位时间内的QPS、RT。
- 统计单位时间内的PV、 UV等。

操作步骤请参见：

- 日志监控概览
- 云监控管理日志监控

10.10 ARMS消费

业务实时监控服务 ARMS 是一款端到端一体化实时监控解决方案的PaaS级阿里云产品，如果您已经开通了阿里云日志服务与ARMS云产品，即可实现ARMS与LogHub无缝对接。

操作步骤：

1. 准备数据源。 配置日志服务采集日志，将LogHub作为ARMS的数据源。
2. 为ARMS配置*RAM*授权，允许ARMS读取LogHub的日志数据。
3. 配置*ARMS*接入*LogHub*数据源。

完成以上步骤后，ARMS已成功同步您的LogHub日志数据，您可以参考快速接入自定义监控，为您的日志数据创建ARMS自定义监控任务。

11 服务监控

11.1 监控日志服务

日志服务支持通过以下方式实现日志监控：

- 通过云监控
  - 日志库（Logstore）级读写
  - Agent（Logtail）日志收集
- 控制台
  - 实时订阅消费（Spark Streaming、Storm、Consumer Library）当前点位
  - 投递 OSS/MaxCompute 状态

其中，云监控方式即通过阿里云云产品云监控完成，控制台方式监控日志请参见控制台 消费组状态、管理日志投递任务和设置告警。

云监控配置步骤

> **说明：**
> 使用子账号配置云监控需要先授权。详细内容请参考 云监控授权文档。

1. 登录日志服务管理控制台。
2. 选择所需的项目，单击项目名称。
3. 选择所需的日志库并单击监控列下的图标打开云监控。

   您也可以通过云监控管理控制台左侧导航栏中的**云服务监控 > 日志服务**进入监控配置页面。

   在云监控中对日志数据进行监控，详细信息请参考云监控中的日志监控。

图 11-1: 云监控中的日志监控



监控项含义

请参考云监控指标。

设置报警规则

在监控图表页面右上角单击监控图表，设置关联资源、报警规则和通知方式。详细步骤请参考设置云监控告警规则。

11.2 服务日志

11.2.1 简介

阿里云日志服务提供服务日志功能，支持记录Project内的用户操作日志、消费组监控日志、Logtail错误日志、Logtail采集日志和计量日志。开通后自动创建以internal- 开头的5个Logstore用于保存这些日志。您可以通过查询分析等操作实时掌握日志服务的使用状况、提高运维效率。

限制说明

- 专属日志库用于存入日志服务产生的日志，因此**不支持写入其他数据**。查询、统计、报警、流式消费等其他功能没有限制。
- 不支持RAM用户开通，请使用主账号操作。
- Project产生的日志只能保存至同一地域的Project中，不支持跨地域操作。
- 日志服务记录日志功能产生的所有日志数据遵循日志服务的计费策略，即按量计费，且按月提供免费额度。计费说明请查看计费方式。
- 开启服务日志功能后，您可以在修改Project时关闭该功能。关闭功能后停止写入服务日志，但不删除历史日志，可能会产生少量费用。如果您需要删除历史日志，可以直接删除保存日志的Logstore。

应用场景

- **查看计量数据**

  用户开通日志服务后，日志服务每个小时统计一次用户每个Logstore的日志和索引占用的存储空间、上一次统计之后用户的读写次数和索引流量等和计费相关的数据。开通服务日志后，计量日志将会实时保存到一个单独的Logstore中，用户可以了解账户的存储量和消费等信息。

- **判断Shard写入和消费是否均衡**

  您可以通过预定义的仪表盘对比Shard数据写入和消费变化趋势来判断Shard写入和消费是否均衡。

  如果Project中存在多个Logstore，不同的Logstore可能存在重复的Shard。因此如果需要查看某个Logstore的Shard写入分布，请在仪表盘左上角的过滤条件增加Project和Logstore作为过滤条件。

- **API请求状态监控**

  用户的所有操作如日志写入、消费、创建Project和Logstore等，都是通过API请求的方式。用户的每次操作都对应internal-operation_log 这个Logstore中的一条日志。如果请求失败，则日志的Status字段为一个大于200的整数如404。因此，用户可以通过监控 Status>200的日志条数来监控API请求是否正常。

- **查看Logtail的状态**

服务日志开通后默认创建两个Logtail相关的仪表盘，分别为异常监控和采集的数据统计。通过异常监控这个仪表盘，用户可以看到Logtail有哪些异常状况出现，如日志解析失败，正则表达式不匹配等。

11.2.2 配置步骤

您可以在创建或修改Project时开启记录日志功能，日志服务会将当前Project产生的所有日志保存到您指定的Project中。该功能默认为关闭状态。

前提条件

1. 已开通日志服务。
2. 已使用主账号登录日志服务控制台。

开通运行日志

1. RAM授权。

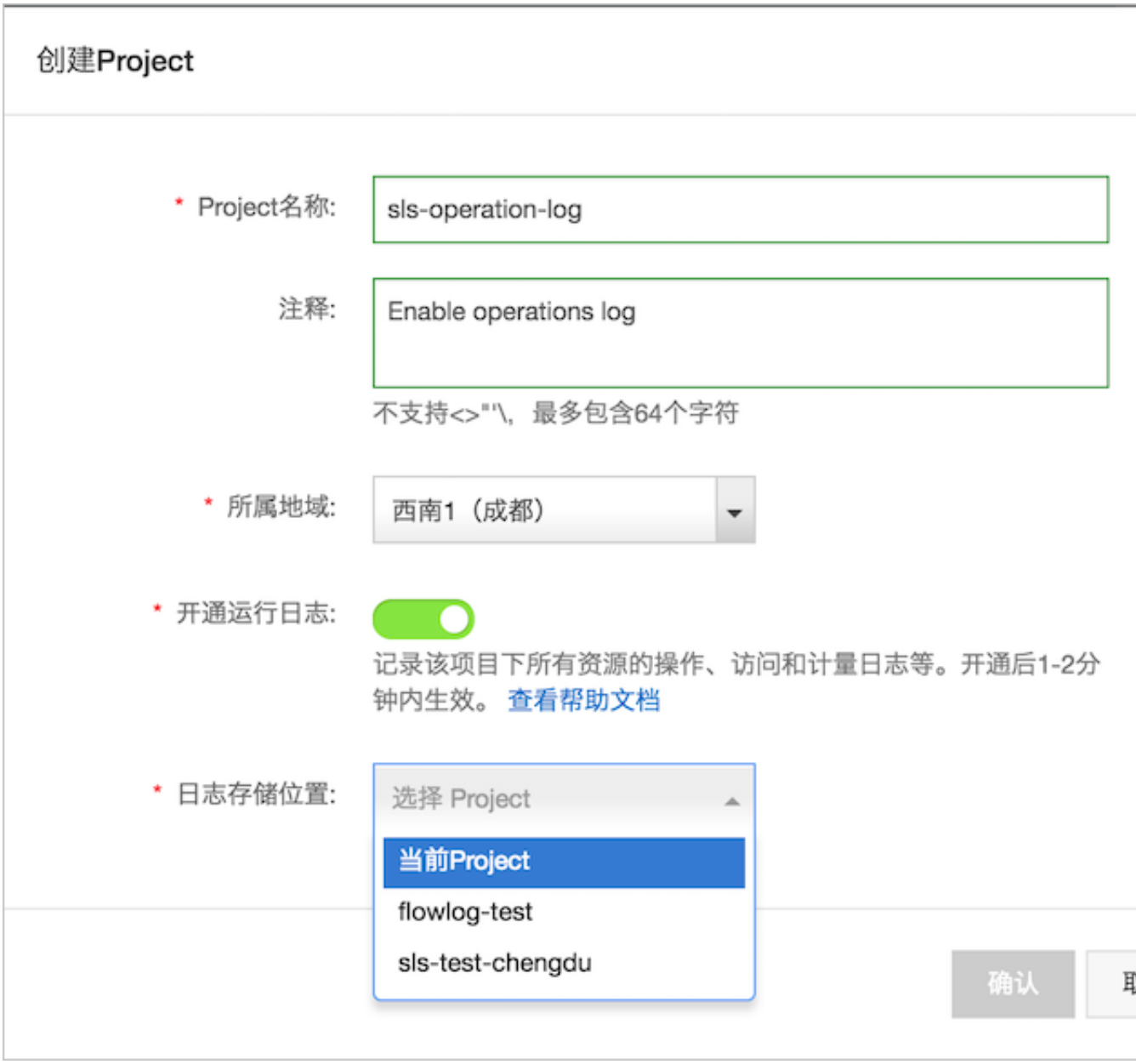
   日志服务需要主账户授权才能将运行日志写入到您指定的Project。单击快捷授权完成授权。
2. 登录日志服务控制台首页，单击右上角的创建**Project**。

   对于已经创建的Project，可以单击修改，在修改**Project**页面中开通运行日志功能。
3. 在弹出页面中，填写**Project**名称等信息。
4. 选择所属地域。
5. 选择日志存储位置，点击确定创建Project同时开通运行日志。

   开通运行日志之后，当前项目下所有资源的操作日志、访问日志和计量日志都会保存到该Project中，您可以选择当前Project或当前地域下其他已创建的Project。

   配置完成后，若您需要修改保存日志的Project，请在修改**Project**步骤中修改，详细说明请参考操作*Project*。

图 **11-2:** 开通运行日志



6. 单击确认。

   您的运行日志功能已成功开通。

7. 在**Project**列表页面单击操作列下的运行日志可以查看保存运行日志的Logstore列表。

   日志服务会在您指定的Project中为您创建5个Logstore，分别用来存储用户操作日志、消费组日志、Logtail告警日志、Logtail采集日志和计量日志。同时创建5个默认仪表盘，分别用来展示访问数据、计量数据、Logtail采集信息、错误信息和消费组信息。

关闭运行日志

1. 主账号登录日志服务控制台，单击操作列的修改。
2. 在弹出页面中关闭开通运行日志功能开关。

   已创建的Logstore、仪表盘和日志信息不会自动删除，若您不再需要该部分数据，可以手动删除。

   图 **11-3:** 关闭运行日志



日志类型与日志字段

关于日志服务的日志类型与字段说明，请参考日志类型。

默认配置

表 11-1: 默认配置

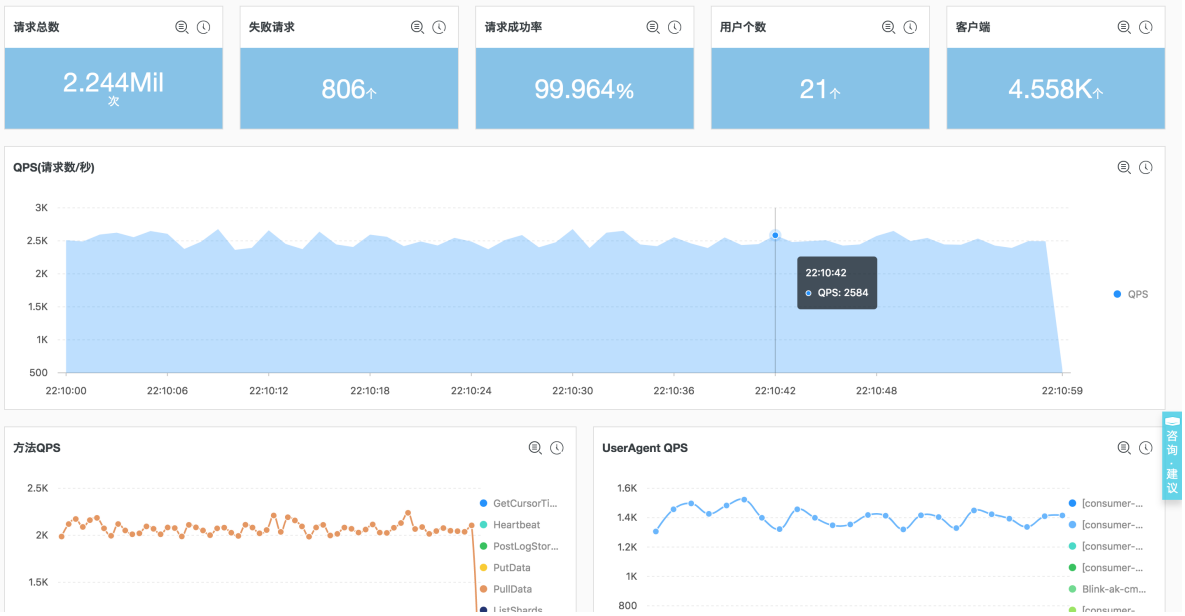
默认配置项	配置内容
Logstore	默认为您创建以下5个Logstore。当前Project产生的所有日志数据都会被分类保存到特定的Logstore中。 • `internal-operation_log`：用于保存用户操作日志。 • `internal-consumergroup_log`：用于保存消费组日志。 • `internal-logtail_alarm`：用于保存Logtail告警日志。 • `internal-logtail_profile`：用于保存Logtail采集日志。 • `internal-metering`：用于保存计量日志。 关于日志类型和字段，请查看日志类型。
地域	日志服务日志仅支持将产生的日志信息保存至当前地域的Project中。
Shard	每个Logstore默认创建2个Shard，并开启自动分裂*Shard* 功能。
日志存储时间	默认保存30天，您也可以自定义设置保存时间。详细步骤请参考操作*Logstore*。
索引	默认为采集到的所有日志数据开启索引。如果您没有查询分析和设置告警等需求，您可以在**查询**页面关闭索引。
仪表盘	默认为您创建5个仪表盘，分别为： • internal_operation_statistics • internal_metering • internal_logtail_statistics • internal_logtail_monitoring • internal_consumer_group_monitoring 关于仪表盘的更多信息，请参考默认仪表盘。

默认仪表盘

- internal_operation_statistics

展示所有的用户访问和操作信息，如QPS、请求延时等信息。包括API请求, 控制台操作等所有请求。

图 11-4: internal_operation_statistics



- internal_metering

展示用户计量相关的数据，如使用的存储量、索引流量、读写次数、读写流量等。

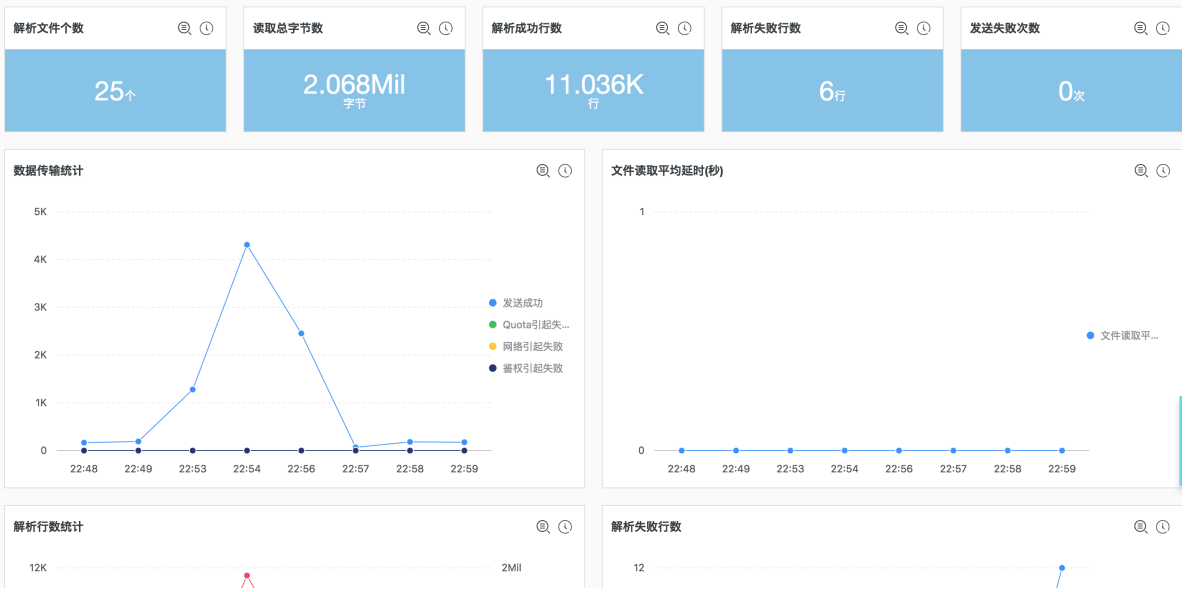
图 11-5: internal_metering



- internal_logtail_statistics

展示了Logtail数据采集相关的统计信息。

图 **11-6: internal_logtail_statistics**



- internal_logtail_monitoring

  展示Logtail所有的错误告警信息，便于用户实时监控Logtail的健康状态。

  图 **11-7: internal_logtail_monitoring**



- internal_consumer_group_monitoring

  展示和消费组相关的统计，包括分Shard的消费数据、消费的落后时间、消费组列表等。

图 11-8: internal_consumer_group_monitoring



11.2.3 日志类型

日志类型

日志来源	Logstore	说明
用户操作日志	internal-operation_log	用户的所有API访问和操作日志。用户所有读写请求，包括控制台，消费组等所有客户端发送的请求。
消费组日志	internal-consumergroup_log	消费组的快照信息，2分钟上报一次。
Logtail告警日志	internal-logtail_alarm	Logtail的错误日志。 Logtail遇到的错误。每30秒上报一次，30秒内重复出现的错误类型只记录错误总和，错误Message随机选择一条。
Logtail采集日志	internal-logtail_profile	Logtail文件采集统计信息。 Logtail采集数据统计信息。10分钟生成一条。
计量日志	internal-metering	用户计量相关统计。

日志来源	Logstore	说明
		Logstore粒度的用户所有操作消费的流量，使用的存储空间，请求次数等。每小时统计一次。

用户操作日志

用户操作日志可以根据Method字段分为读数据，写数据和其他元数据三种操作。

分类	方法
读数据	• GetLogStoreHistogram • GetLogStoreLogs • GetDataMeta • GetData • PullData • GetCursor • GetCursorTime
写数据	• PostLogStoreLogs • PutData • WebTracking
元数据操作	CreateProject, DeleteProject等其他方法。

公共字段

所有操作的公共字段如下表所示。

字段	描述	示例
APIVersion	API 版本。	0.6.0
InvokerUid	执行操作的用户的账户id。	1759218115323050
NetworkOut	通过公网入口读取的流量，单位为字节。	10
Latency	请求延时，单位为微秒。	123279
LogStore	Logstore的名称。	logstore-1
Method	执行方法。	GetLogStoreLogs
Project	Project名称。	project-1

字段	描述	示例
NetOutFlow	读取的流量，单位为字节。	120
RequestId	请求ID。	8AEADC8B0AF2FA2592C9509E
SourceIP	发送请求的客户端IP。	1.2.3.4
Status	请求响应状态码。	200
UserAgent	客户端用户代理。	sls-java-sdk-v-0.6.1

读操作字段

读请求特有的字段如下表所示：

字段	描述	示例
BeginTime	请求开始时间，Unix时间戳。	1523868463
DataStatus	请求响应数据状态。Complete，OK，Unknown等。	OK
EndTime	请求结束时间，Unix时间戳。	1523869363
Offset	GetLog请求偏移行数。	20
Query	原始查询语句。	UserAgent: [consumer-group-java]*
RequestLines	期望返回行数。	100
ResponseLines	返回结果行数。	100
Reverse	是否按日志时间戳逆序返回日志，其中： • 1 表示逆序。 • 0表示顺序。 默认值为 0。	0
TermUnit	搜索语句中包含的词项个数。	0
Topic	读取的主题名称。	topic-1

写操作字段

写请求特有的字段如下表所示：

字段	描述	示例
InFlow	原始写入字节数。	200
InputLines	请求写入行数。	10
NetInflow	压缩之后的写入字节数。	100
Shard	写入的Shard Id。	1
Topic	数据写入的topic。	topic-1

消费组快照日志

字段	描述	示例
consumer_group	消费组名称。	consumer-group-1
fallbehind	当前消费位置距离最新写入日志的落后时间，单位为秒。	12345
logstore	日志库名称。	logstore-1
project	项目名称。	project-1
shard	Shard Id。	1

Logtail告警日志

字段	描述	示例
alarm_count	采样窗口内告警次数。	10
alarm_message	触发告警的原始日志采样。	M_INFO_COL,all_status_monitor,T22380,0,2018-04-17 10:48:25.0,AY66K,AM5,2018-04-17 10:48:25.0,2018-04-17 10:48:30.561,i-23xebl5ni.1569395.715455,901,00789b
alarm_type	告警类型。	REGISTER_INOTIFY_FAIL_ALARM
logstore	日志库名称。	logstore-1
source_ip	Logtail运行的机器IP地址。	1.2.3.4
os	操作系统，Linux或Windows等。	Linux
project	项目名称。	project-1

字段	描述	示例
version	Logtail版本。	0.14.2

Logtail采集日志

Logtail采集日志可以根据字段`file_name`分为两类，一类是针对单个文件的采集统计信息，一类是Logstore级别的统计信息，即`file_name`为`logstore_statistics`的部分。当`file_name`为`logstore_statistics`时，文件相关的字段如`file_dev`、`file_inode`无意义。字段说明如下表所示。

字段	描述	示例
logstore	日志库名称。	logstore-1
config_name	采集配置名。由##配置版本号##*projectName*$配置名称组成的全局唯一配置名。	##1.0##project-1$logstore-1
error_line	引起错误的原始日志。	M_INFO_COL,all_status_monitor,T22380,0,2018-04-17 10:48:25.0,AY66K,AM5,2018-04-17 10:48:25.0,2018-04-17 10:48:30.561,i-23xebl5ni.1569395.715455,901,00789b
file_dev	该日志文件的device ID。	123
file_inode	该日志文件的inode。	124
file_name	日志文件完整路径或者或者`logstore_statistics`。	/abc/file_1
file_size	当前文件大小，单位为字节。	12345
history_data_failures	历史处理失败次数。	0
last_read_time	窗口内最近的读取时间，Unix时间戳。	1525346677
project	项目名称。	project-1
logtail_version	Logtail版本。	0.14.2
os	操作系统。	Windows
parse_failures	窗口日志解析失败的行数。	12

字段	描述	示例
read_avg_delay	窗口内平均每次读取时当前偏移量与文件大小差值的平均值。	65
read_count	窗口内日志读取次数。	10
read_offset	当前读取到文件偏移位置，单位为字节。	12345
regex_match_failures	正则表达式匹配失败次数。	1
send_failures	窗口内发送失败的次数。	12
source_ip	Logtail运行机器IP地址。	1.2.3.4
succeed_lines	处理成功的Log行数。	123
time_format_failures	日志时间匹配失败次数。	122
total_bytes	读取的总字节数。	12345

如下字段只有在`file_name`为`logstore_statistics`时才会存在：

字段	描述	示例
send_block_flag	窗口结束时发送队列是否阻塞。	false
send_discard_error	窗口内因数据异常或无权限导致丢弃数据包的个数。	0
send_network_error	窗口内因网络错误导致发送失败的数据包个数。	12
send_queue_size	窗口结束时当前发送队列中未发送数据包数。	3
send_quota_error	窗口内因quota超限导致发送失败的数据包个数。	0
send_success_count	窗口内发送成功的数据包个数	12345
sender_valid_flag	窗口结束时该Logstore的发送标志位是否有效，其中： • true表示正常。 • false表示可能因为网络错误或quota错误而被禁用。	true

字段	描述	示例
max_send_success_time	窗口内发送成功数据的最大时间。Unix时间戳。	1525342763
max_unsend_time	窗口结束时发送队列中未发送数据包的最大时间，队列空时为0。Unix时间戳。	1525342764
min_unsend_time	窗口结束时发送队列中未发送数据包的最小时间，队列空时为0。Unix时间戳。	1525342764

计量日志

字段	描述	示例
begin_time	统计窗口开始时间，Unix时间戳。	1525341600
index_flow	统计窗口内索引流量，单位为字节。	12312
inflow	统计窗口内写入流量，单位为字节。	12345
logstore	Logstore名称。	logstore-1
network_out	统计窗口内流出到公网流量，单位为字节。	12345
outflow	统计窗口内读取流量，单位为字节。	23456
project	项目名称。	project-1
read_count	统计窗口内读操作次数。	100
shard	统计窗口内平均使用shard个数。	10.0
storage_index	统计时间点Logstore内索引总存储用量，单位为字节。	10000000
storage_raw	统计时间点Logstore内日志总存储用量，单位为字节。	20000000
write_count	统计窗口内写操作次数。	199

11.3 云监控方式

11.3.1 云监控指标

监控数据入口请参考监控日志服务。

1. 写入/读取流量
   - 含义：每个日志库（Logstore）写入、以及读取实时情况。统计该Logstore通过ilogtail和SDK、API等读写实时流量，大小为传输大小（压缩情况下为压缩后），每分钟统计一个点。
   - 单位：Bytes/min
2. 原始数据大小
   - 含义：每个Logstore写入数据原始大小，即压缩前的大小。
   - 单位：Byte/min
3. 总体QPS
   - 含义：所有操作QPS，每分钟统计一个点。
   - 单位：Count/Min
4. 操作次数
   - 含义：统计用户的各种操作对应的QPS，每分钟统计一个点。
   - 单位：次/分钟（Count/Min)
   - 所有的操作包括：
     - 写入操作：
       - PostLogStoreLogs ：0.5API以后版本接口。
       - PutData ： 0.4 API以前版本接口。
     - 根据关键字查询：
       - GetLogStoreHistogram: 查询关键字分布情况，0.5API以后版本接口。
       - GetLogStoreLogs: 查询关键字命中日志，0.5API以后版本接口。
       - GetDataMeta ： 同GetLogStoreHistogram，为0.4API以前版本接口。
       - GetData : 同GetLogStoreLogs，为0.4API以前版本接口。
     - 批量获取数据：
       - GetCursorOrData：该操作包含了获取Cursor和批量获取数据两种方法。

- ■ ListShards：获取一个Logstore下所有的Shard。

— List操作：

- ■ ListCategory：同ListLogStores，为0.4API以前版本接口。
- ■ ListTopics：遍历一个Logstore下所有的Topic。

5. 服务状态

- 含义：该视图统计用户的各种操作返回的HTTP 状态码对应的QPS，方便用户根据错误的返回码来判断操作异常，及时调整程序。
- 各状态码:

— 200：为正常的返回码，表示操作成功。

— 400：错误的参数，包括Host、Content-length、APIVersion、RequestTimeExpired、查询时间范围，Reverse，AcceptEncoding、AcceptContentType、Shard、Cursor、PostBody、Paramter和ContentType等方面的错误。

— 401：鉴权失败，包括AccessKeyId不存在、签名不匹配或者签名账户没有操作权限，请到日志服务控制台上查看Project权限列表，是否包含了该AK。

— 403: 超过预定Quota，包括能够创建的Logstore个数、Shard总数、以及读写操作的每分钟限额，请根据返回的Message判断发生了哪种错误。

— 404：请求的资源不存在，包括Project、 Logstore、Topic 、User等资源。

— 405：错误的操作方法，请检查请求的URL路径。

— 500：服务端错误，请重试。

— 502：服务端错误，请重试。

6. 客户端解析成功流量

- 含义：Logtail收集成功的日志大小，为原始数据大小。
- 单位：字节

7. 客户端（Logtail）解析成功行数

- 含义：Logtail收集成功的日志的行数。
- 单位: 行

8. 客户端解析失败行数

- 含义：Logtail收集日志过程中，采集出错的行数大小，如果该视图有数据，表示有错误发生。

- 单位：行

9. 客户端错误次数
   - 含义：Logtail收集日志过程中，出现所有收集错误的IP总数。
   - 单位：次
10. 发生客户端错误机器数
    - 含义：Logtail收集日志过程中，出现收集错误的告警客户端数目。
    - 单位：个
11. 错误IP统计（Count/5min)
    - 含义：分类别展示各种采集错误发生的IP数，各种错误包括：
      - LOGFILE_PERMINSSION_ALARM:没有权限打开日志文件。
      - SENDER_BUFFER_FULL_ALARM：数据采集速度超过了网络发送速度，数据被丢弃。
      - INOTIFY_DIR_NUM_LIMIT_ALARM（INOTIFY_DIR_QUOTA_ALARM）：监控的目录个数超过了3000个，请把监控的根目录设置成更低层目录。
      - DISCARD_DATA_ALARM：数据丢失，因为数据时间在系统时间之前15分钟，请保证新写入日志文件的数据是在15分钟之内的。
      - MULTI_CONFIG_MATCH_ALARM：有多个配置在收集同一个文件，Logtail会随机选择一个配置进行收集，另一个配置则收集不到数据。
      - REGISTER_INOTIFY_FAIL_ALARM：注册inotify事件失败，具体原因请查看Logtail日志。
      - LOGDIR_PERMINSSION_ALARM：没有权限打开监控目录。
      - REGEX_MATCH_ALARM：正则式匹配错误，请调整正则式。
      - ENCODING_CONVERT_ALARM：转换日志编码格式时出现错误，具体原因请查看Logtail日志。
      - PARSE_LOG_FAIL_ALARM：解析日志错误，一般是行首正则表达式错误或单条日志超过512KB导致的日志分行错误，请查看Logtail日志确定原因，如行首正则表达式错误请调整配置。
      - DISCARD_DATA_ALARM：丢弃数据，Logtail发送数据到日志服务失败且写本地缓存文件失败导致，可能的原因是日志文件产生较快但写磁盘缓存文件较慢。

- SEND_DATA_FAIL_ALARM：解析完成的日志数据发送日志服务失败，请查看Logtail日志发送数据失败相关ErrorCode和ErrorMessage，常见的错误有服务端Quota超限、客户端网络异常等。
- PARSE_TIME_FAIL_ALARM：解析日志time字段出错，Logtail根据正则表达式解析出来的time字段按照时间格式配置无法解析成功，请修改配置。
- OUTDATED_LOG_ALARM：Logtail丢弃历史数据，请保证当前写入日志数据的时间与系统时间相差在5分钟以内。

- 请根据具体错误找到出错IP，登录机器查看/usr/logtail/ilogtail.LOG，分析错误原因。

11.3.2 设置云监控告警规则

日志服务支持通过云监控设置报警，当服务状态符合设置的报警规则时发送报警短信或邮件。您可以通过配置云监控中的日志监控报警规则，对日志收集状态、Shard资源使用状态等异常状态进行监控。

操作步骤

在云监控控制台**云服务监控 > 日志监控**页面，找到需要设置监控报警的Logstore，单击其右侧的**报警规则**。在报警规则右上角单击**创建报警规则**。

1. 关联资源。
   a. 选择产品。此处请选择**日志服务**。
   b. 选择资源范围。

      您可以选择**全部资源**、**应用分组**和**Project维度**。

      - 资源范围选择**全部资源**，则产品下任何实例满足报警规则描述时，都会发送报警通知。
      - 选择**应用分组**，则应用分组下任何实例满足报警规则表述时，都会发送报警通知。
      - 选择**Project维度**，则选中的实例满足报警规则描述时，才会发送报警通知。

   c. 选择**地域**。
   d. 选择**Project**和**Logstore**。您可以选择一个及以上的Project和Logstore。

图 11-9: 关联资源



2. 设置报警规则。

   您可以设置一条或多条报警规则。

   a. 填写规则名称。

   b. 填写规则描述。

      您需要在此处定义您的监控策略，选择需要的监控项目，并为其设定阈值。超出该值后云监控会发送报警通知。

      各个监控项的含义请参考云监控指标，统计方法请参考监控日志服务。

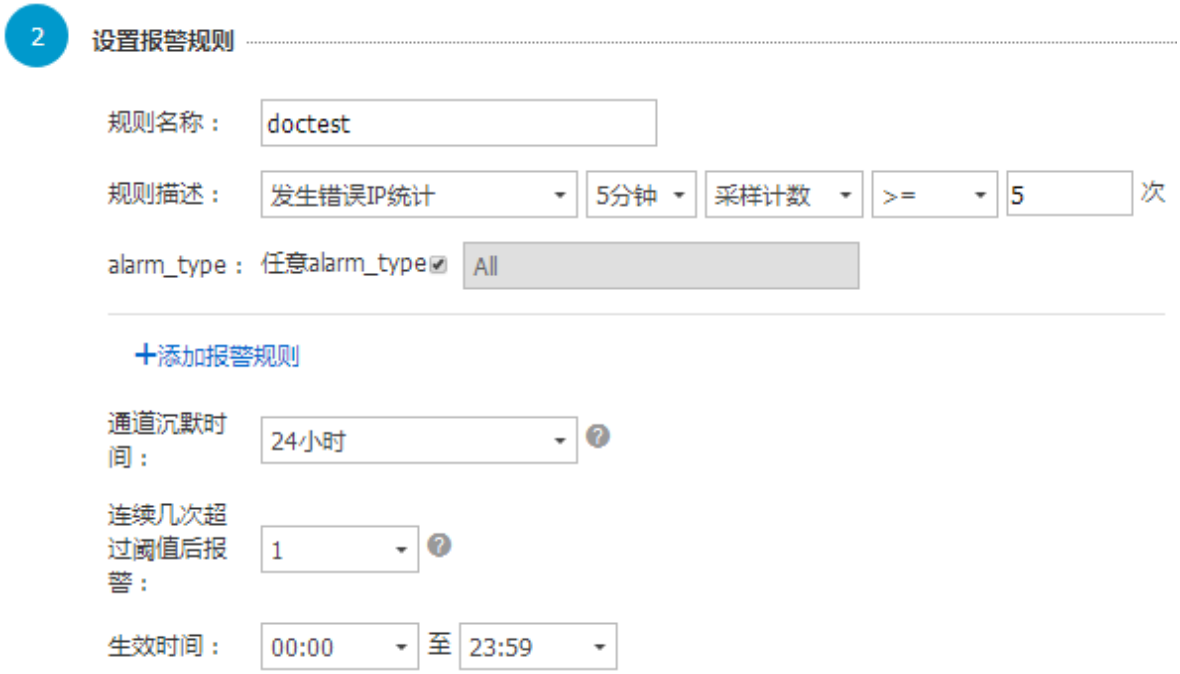
   c. 选择**alarm_type**。默认为任意**alarm_type**。

   d. 设置通道沉默时间。指报警发生后如果未恢复正常，间隔多久重复发送一次报警通知。

   e. 连续几次超过阈值后报警。即连续几次报警的探测结果符合您设置的规则描述，才会触发报警。

   f. 生效时间。为您的监控策略选择生效时间，设定后仅在该时段内执行监控报警策略。

图 **11-10:** 设置报警规则



3. 配置通知方式。

   a. 通知对象。以联系人组级别发送通知。

   b. 报警级别。您可以按照需要选择**Warning**和**Info**级别，不同级别通知方式不同。

   c. 邮件主题和邮件备注。邮件主题默认为产品名称+监控项名称+实例ID。

   d. 报警回调。填写公网可访问的URL，云监控会将报警信息通过POST请求推送到该地址，目前仅支持HTTP协议。

图 11-11: 通知方式



配置完成后单击**确认**，完成监控策略配置。

示例

监控Logtail日志收集状态

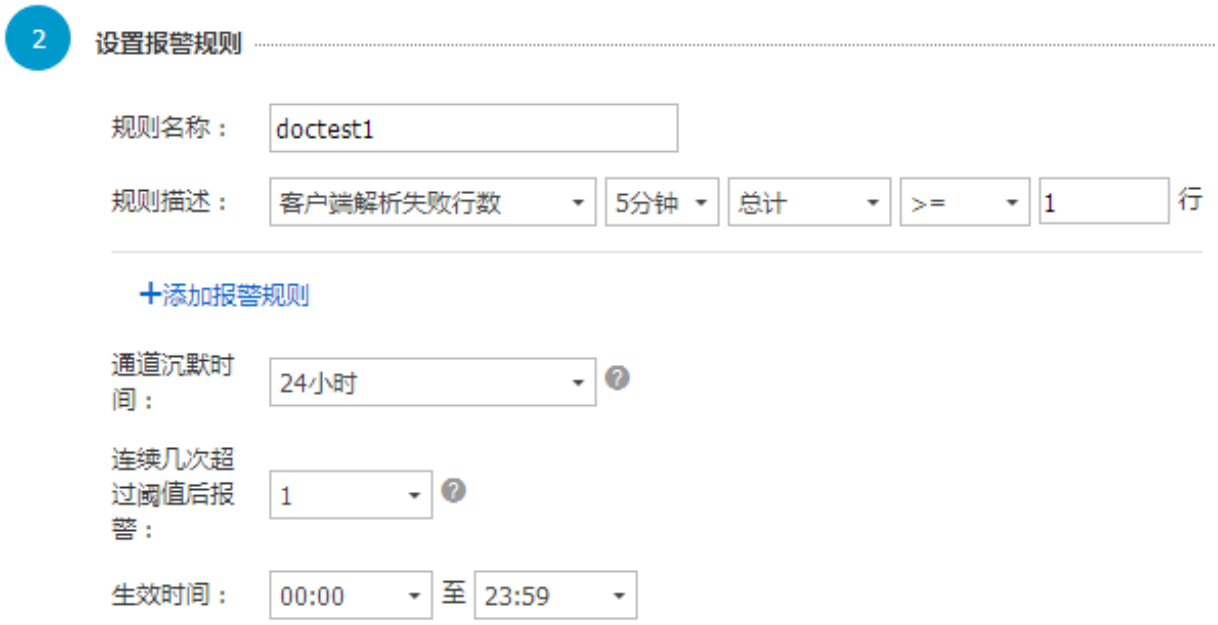
Logtail客户端在运行过程中，可能会因设置不正确产生错误，例如某些日志格式不匹配、一个日志文件被重复收集等。为了及时发现这种情况，您可以对客户端解析失败行数、客户端错误次数等指标进行监控，以便及时发现这类问题。

该监控的监控规则设置如下：

您可以根据需要选择**客户端解析失败行数**或**客户端错误次数**选项，并配置统计周期、统计方法等规则项。除此之外，还可以根据Logtail其他错误项进行报警，第一时间发现各类日志收集过程中发现的问题。

下图示例表示：五分钟内客户端解析失败行数达到一行以上即发送报警，持续24H监控。

图 **11-12:** 监控**Logtail**日志收集状态



监控**Shard**资源使用状态

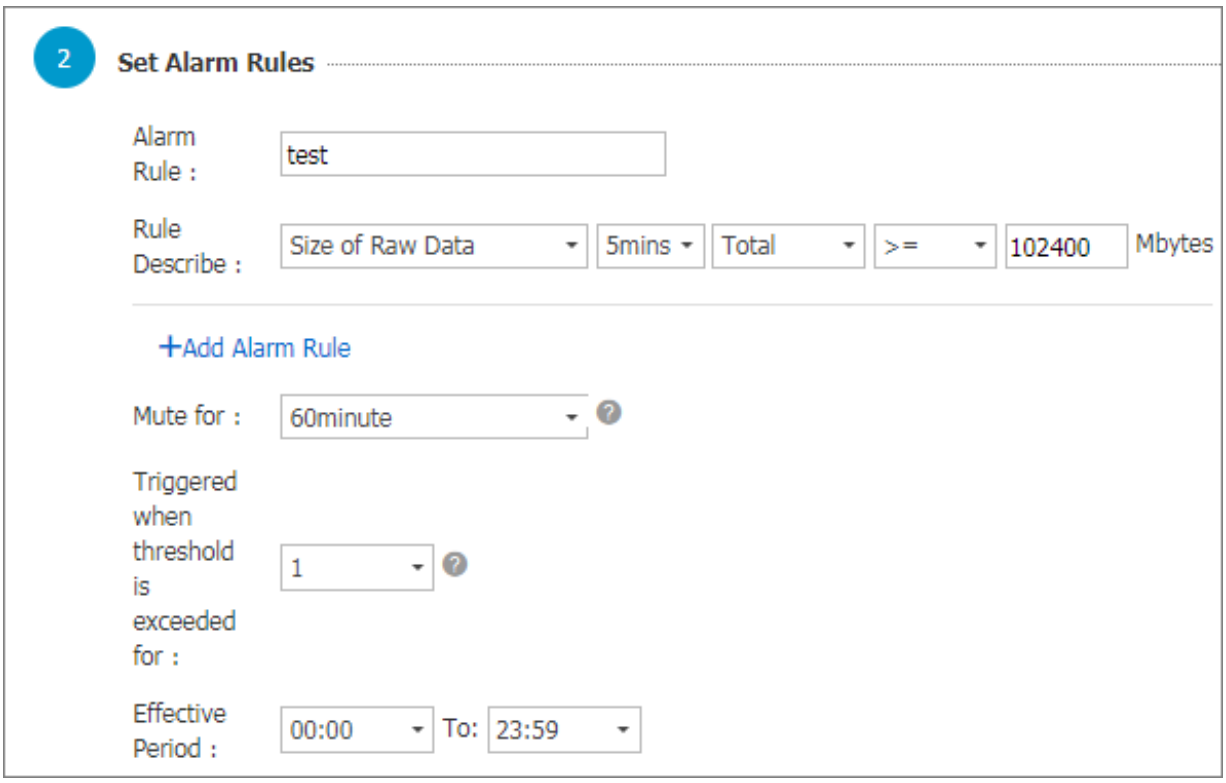
Logstore下每个Shard提供5MB/s (500次/s) 写入能力，这个数值对于大部分用户而言都是足够的，在超过时日志服务会尽可能去服务（非拒绝）您的请求，但在高峰期间不保证超出部分的可用性。您可以设置Logstore出入流量报警以检测该情况。如果您的日志量非常大，需要添加更多Shard，请及时在控制台中进行调整。

设置Logstore流量报警规则如下。

方案**1**：设置流量预警

设置**规则名称**，并配置**规则描述**为**原始数据大小**。您可以在此处设置统计周期和统计方法，如需超过100GB/5Min后进行报警，请设置**5分钟**、**总计**、**> = 、102400**，表示5分钟内总计流量超出102400Mbytes时进行报警。

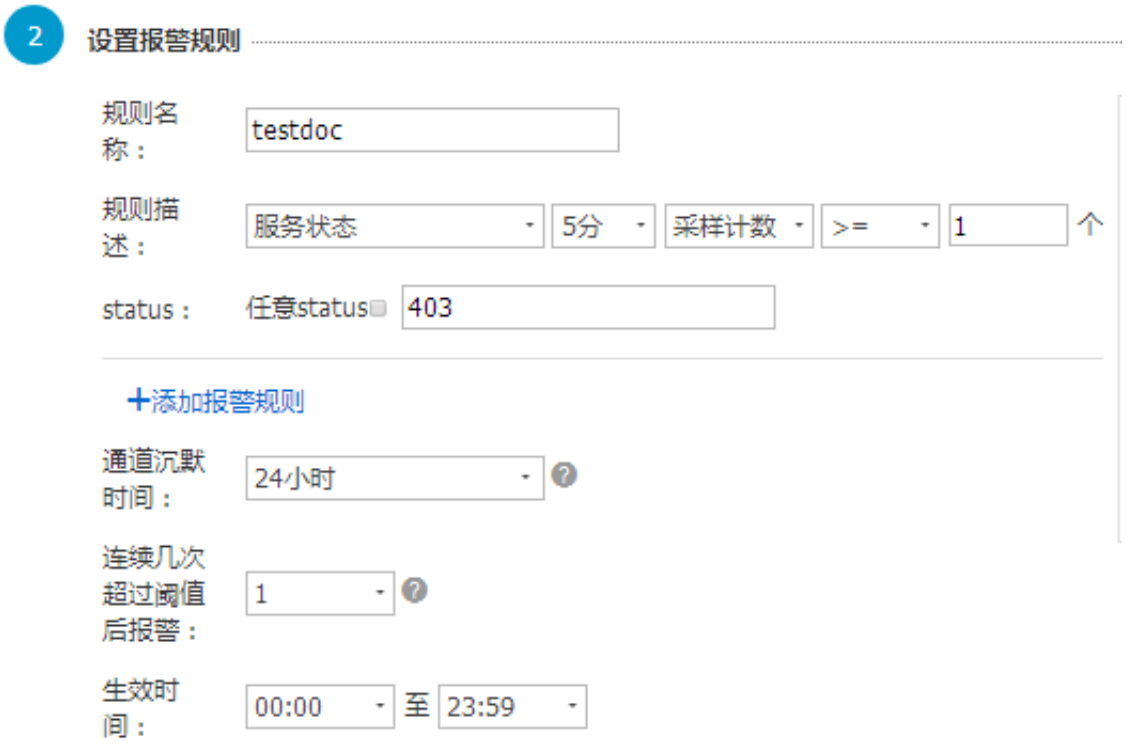
图 **11-13:** 设置流量预警



方案2：设置服务状态报警

设置规则名称，并配置规则描述为服务状态。您可以在此处设置统计周期和统计方法，如您需要在5分钟内出现1个以上403服务状态时收到报警，请参考下图配置。

图 **11-14:** 设置服务状态报警



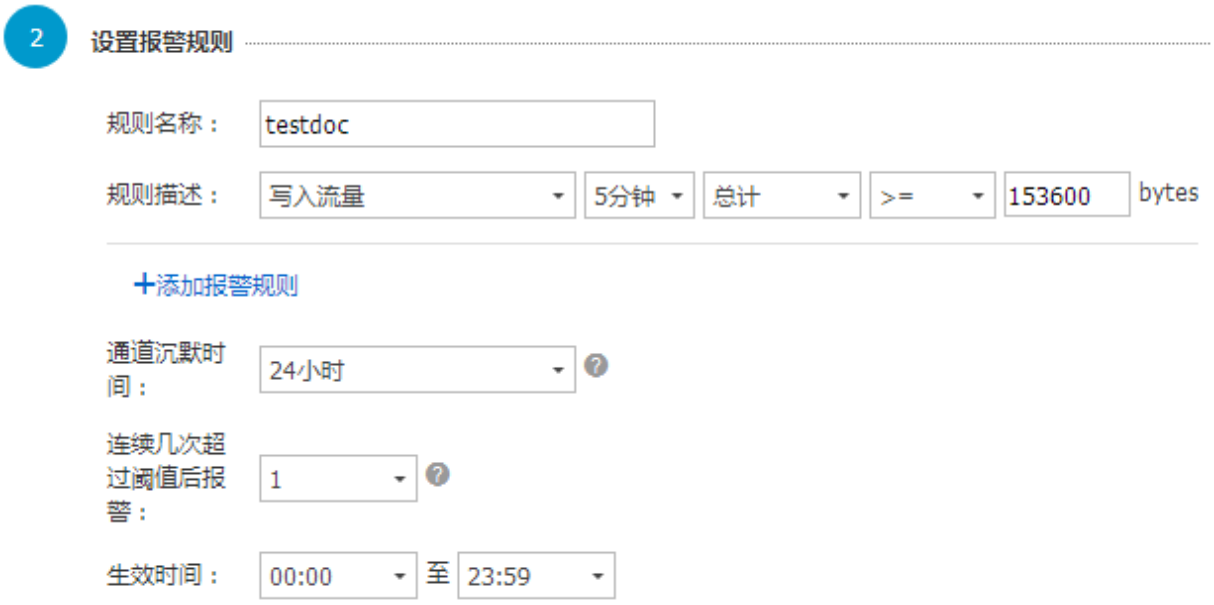
监控Project的写入流量

每个Project默认写入限制为30GB/分钟（原始数据大小），这个数值主要目的是为了保护用户因程序错误产生大量日志，在一般场景中对于大部分用户都是足够的。如果您的日志量非常大，可能会超过限制，可以通过工单联系我们调整大这个数值。

您可以按照以下示例设置Project Quota的监控策略。

该示例表示：当您的五分钟写入流量超过了150GB时，为您发送提醒。

图 11-15: 监控Project的写入流量



12 数据投递

12.1 简介

将日志源接入日志服务后，日志服务开始实时采集日志，并提供控制台或SDK/API方式的日志消费和日志投递功能。日志服务采集到LogHub的日志数据，可以实时投递至OSS、Table Store、MaxCompute等存储类阿里云产品，您只需要在控制台配置即可完成，同时，LogShipper提供完整状态API与自动重试功能。

应用场景

对接数据仓库

日志数据来源

日志服务的日志投递功能所投递的日志数据来源于日志服务采集到LogHub的日志。这部分日志生成之后，被日志服务实时采集并投递至其他云产品中进行存储与分析。

日志投递目标

- OSS（大规模对象存储）：
  - 投递流程
  - OSS 上格式可以通过 Hive 处理，推荐 E-MapReduce
- Table Store（NoSQL 数据存储服务）：
  - 操作步骤
- MaxCompute (大数据计算服务)：
  - 直接投递-通过日志服务投递日志到*MaxCompute*
  - 通过DataWorks数据集成投递-操作步骤
  - 通过DataWorks数据集成投递-操作步骤

12.2 投递日志到OSS

12.2.1 投递流程

日志服务可以把Logstore中的数据自动归档到OSS，以发挥日志更多的效用。

- OSS 数据支持自由设置生命周期，可以对日志进行长期存储。
- 可以通过自建程序和更多系统（如E-MapReduce）消费OSS数据。

功能优势

通过日志服务投递日志数据到OSS具有如下优势：

- 操作简单。仅需在控制台上做简单配置即可将日志服务Logstore的数据同步到OSS。
- 效率提升。日志服务的日志收集过程已经完成不同机器上的日志集中化，无需重复在不同机器上收集日志导入OSS。
- 便于管理。投递日志到OSS可以充分复用日志服务内的日志分类管理工作。用户可让日志服务不同项目（Project）、不同类型（Logstore）的日志自动投递到不同的OSS Bucket目录，方便管理OSS数据。

前提条件

1. 开通日志服务，创建Project和Logstore，并成功采集到日志数据。
2. 开通OSS，在日志服务Project所在的地域创建Bucket。
3. 开通访问控制RAM。
4. 日志服务Project和OSS的Bucket必须位于相同Region，不支持跨Region投递数据。

操作步骤

1 访问控制（RAM）授权

开启投递任务之前，您需要为日志服务授权，允许日志数据写入OSS。

单击快捷授权，在弹出页面中单击**同意授权**。成功授权后，日志服务具备OSS的数据写入权限。

> **说明：**
>
> - 修改授权策略、跨阿里云账号配置投递任务，请参考*RAM*授权。
> - 不涉及跨阿里云账号时，子账号配置投递任务请参考授权*RAM* 用户，为子账号授予权限。

2 配置OSS投递规则

1. 登录日志服务控制台。
2. 选择所需的项目，单击项目名称。
3. 选择所需的日志库，单击日志投递列下的**OSS**。
4. 单击**开启投递**，设置 OSS 投递配置并单击**确认**。

   请参考下表设置OSS投递配置。

配置项	说明	取值范围
OSS投递名称	您所创建的投递的名称。	只能包含小写字母，数字，连字符（-）和下划线（_），必须以小写字母和数字开头和结尾，且名称长度为3~63字节。
OSS Bucket	OSS Bucket 名称。	必须是已存在的Bucket名称，且需要保证 OSS 的 Bucket 与日志服务 Project 位于相同 Region。
OSS Prefix	OSS前缀，从日志服务同步到 OSS 的数据将存放到 Bucket 的该目录下。	必须是已存在的OSS Prefix名称。
分区格式	将投递任务创建时间使用%Y，%m，%d，%H，%M等格式化生成分区字符串，以此来定义写到OSS的Object文件所在的目录层次结构，其中斜线/表示一级OSS目录。如下表举例说明OSS Prefix和分区格式如何定义OSS目标文件路径。	格式化参考*strptime API*。
RAM角色	RAM角色ARN及角色名称，用于访问权限控制，OSS Bucket 拥有者创建角色的标识，如何获得ARN请参见步骤1中的查看、修改角色。	如`acs:ram::45643:role/aliyunlogdefaultrole`。
投递大小	自动控制投递任务创建间隔并设置 OSS 的一个 Object 大小（以未压缩计算）上限。	取值范围为5~256，单位为 MB。
存储格式	日志数据投递OSS的存储格式。	支持JSON/Parquet/CSV三种格式，配置细节请点击查看：*JSON格式*、*Parquet格式*、*CSV格式*。
是否压缩	OSS 数据存储的压缩方式。	• none：表示原始数据不压缩。 • snappy：表示使用 *snappy* 算法对数据做压缩，可以减

配置项	说明	取值范围
		少 OSS Bucket 存储空间使用量。
投递时间	投递任务的间隔时长。	取值范围为300~900，默认值300。单位为秒。

图 12-1: 投递日志



图 12-2: 角色ARN



说明：

日志服务在后端并发执行数据投递，如果您的数据量较大，可能会多个投递线程进行服务。每一个投递线程都会根据投递大小、投递时间决定任务生成的频率，当任一条件满足时，投递线程即会创建任务。

分区格式

每个投递任务会写入OSS一个文件，路径格式是`oss:// OSS-BUCKET/OSS-PREFIX/PARTITION-FORMAT_RANDOM-ID`。PARTITION-FORMAT根据投递任务创建时间格式化得到，以创建时间为2017/01/20 19:50:43的投递任务为例，说明分区格式的用法：

OSS Bucket	OSS Prefix	分区格式	OSS文件路径
test-bucket	test-table	%Y/%m/%d/%H/%M	`oss://test-bucket/test-table/2017/01/20/19/50/43_1484913043351525351_2850008`
test-bucket	log_ship_oss_example	year=%Y/mon=%m/day=%d/log_%H%M%s	`oss://test-bucket/log_ship_oss_example/year=2017/mon=01/day=20/log_195043_1484913043351525351_2850008.parquet`
test-bucket	log_ship_oss_example	ds=%Y%m%d/%H	`oss://test-bucket/log_ship_oss_example/ds=20170120/19_1484913043351525351_2850008.snappy`
test-bucket	log_ship_oss_example	%Y%m%d/	`oss://test-bucket/log_ship_oss_example/20170120/_1484913043351525351_2850008`
test-bucket	log_ship_oss_example	%Y%m%d%H	`oss://test-bucket/log_ship_o`

OSS Bucket	OSS Prefix	分区格式	OSS文件路径
			ss_example/2017012019_148491304335152535 1_2850008

使用Hive或MaxCompute等大数据平台分析OSS数据时，如果希望使用Partition信息，可以设置每一层目录上为key=value格式（Hive-style partition）。

例如：`oss://test-bucket/log_ship_oss_example/year=2017/mon=01/day=20/log_195043_1484913043351525351_2850008.parquet`

可以设置三层分区列，分别为：year、mon、day。

日志投递任务管理

在启动 OSS 投递功能后，日志服务后台会定期启动投递任务。您可以在控制台上看到这些投递任务的状态。

通过 日志投递任务管理，您可以：

- 查看过去两天内的所有日志投递任务，了解其状态。投递任务状态可以是“成功”、“进行中”和“失败”。“失败”状态则表示您的投递任务出现了因外部原因而无法重试的错误，需要您参与解决问题。
- 对于创建两天内的投递失败任务，您可在任务列表中查看导致失败的外部原因。修复好这些外部原因后，您可以逐一或者整体重试所有失败任务。

操作步骤：

1. 登录日志服务控制台。
2. 选择所需的项目，单击项目名称。
3. 选择所需的日志库，单击日志投递列下的**OSS**。

   您可以查看任务开始时间、任务结束的时间、接受日志的时间、数据行数、投递任务的状态等信息。

   如果投递任务执行失败，控制台上会显示相应的错误信息，系统会按照策略默认为你重试，您也可以手动重试。

任务重试

一般情况下，日志数据在写入Logstore后的 30 分钟内同步到 OSS。

日志服务默认会按照退火策略重试最近两天之内的任务，重试等待的最小间隔是 15 分钟。当任务执行失败时，第一次失败需要等待 15 分钟再试，第二次失败需要等待 30 分钟（2 乘以 15）再试，第三次失败需要等待 60 分钟（2 乘以 30）再试，以此类推。

如需立即重试失败任务，可以通过控制台单击**重试全部失败任务** 或通过 API/SDK 方式指定任务进行重试。

错误信息

常见失败任务的错误信息如下：

错误信息	错误原因	处理方法
UnAuthorized	没有权限。	请确认以下配置： • OSS 用户是否已创建角色。 • 角色描述的账号 ID 是否正确。 • 角色是否授予 OSS Bucket 写权限。 • role-arn 是否配置正确。
ConfigNotExist	配置不存在	一般是由于删除投递规则导致，如又重新创建了规则，可以通过重试来解决。
InvalidOssBucket	OSS Bucket 不存在。	请确认以下配置： • OSS Bucket 所在 Region 是否与日志服务Project一致。 • Bucket 名称是否配置正确。
InternalServerError	日志服务内部错误。	通过重试解决。

OSS 数据存储

可以通过控制台、API/SDK 或其它方式访问到 OSS 数据。

如使用 Web 管理控制台访问，进入 OSS 服务，选择 Bucket，单击 **文件管理** 即可看到有日志服务投递过来的数据。

更多 OSS 使用细节请参考OSS文档。

Object 地址

```
oss:// OSS-BUCKET/OSS-PREFIX/PARTITION-FORMAT_RANDOM-ID
```

- 路径字段说明
  - OSS-BUCKET、OSS-PREFIX 表示 OSS 的 Bucket 名称和目录前缀，由用户配置，INCREMENTID 是系统添加的随机数。
  - PARTITION-FORMAT定义为%Y/%m/%d/%H/%M，其中%Y，%m，%d，%H，%M分别表示年、月、日、小时、分钟，由本次投递任务的服务端创建时间通过*strptime API*计算得到。
  - RANDOM-ID是一个投递任务的唯一标识。
- 目录的时间含义

  OSS数据目录是按照投递任务创建时间设置的，假设 5 分钟数据投递一次 OSS，2016-06-23 00:00:00 创建的投递任务，它投递的数据是 2016-06-22 23:55 后写入日志服务的数据。如需分析完整的 2016-06-22 全天日志，除了 2016/06/22 目录下的全部 object 以外，还需要检查`2016/06/23/00/` 目录下前十分钟的 Object 是否有包含 2016-06-22 时间的日志。

Object存储格式

- JSON

  请参考*JSON*格式。
- Parquet

  请参考*Parquet*格式。
- CSV

  请参考*CSV*格式。

12.2.2 CSV格式

本文介绍日志服务投递OSS使用CSV存储的相关细节，其它内容请参考投递日志到*OSS*。

CSV存储字段配置

配置页面

您可以在日志服务数据预览或索引查询页面查看一条日志的多个Key-Value，将你需要投递到OSS的字段名（Key）有序填入。

如您配置的Key名称在日志中找不到，CSV行中这里一列值将设置为空值字符串（null）。

图 12-3: 配置项



配置项

配置项	取值	备注
分隔符 delimiter	字符	长度为1的字符串，用于分割不同字段。
转义符 quote	字符	长度为1的字符串，字段内出现分隔符（delimiter）或换行符等情况时，需要用quote前后包裹这个字段，避免读数据时造成字段错误切分。
跳出符 escape	字符	长度为1的字符串，默认设置与quote相同，暂不支持修改。字段内部出现quote（当成正常

配置项	取值	备注
		字符而不是转义符）时需要在quote前面加上escape做转义。
无效字段内容 null	字符串	当指定Key值不存在时，字段填写该字符串表示该字段无值。
投递字段名称 header	布尔	是否在csv文件的首行加上字段名的描述。

更多内容请参考*CSV*标准、*postgresql CSV*说明。

可配置的保留字段

在投递OSS过程中，除了使用日志本身的Key-Value外，日志服务保留同时提供以下几个保留字段可供选择：

保留字段	语义
`__time__`	日志的 Unix 时间戳（是从 1970 年 1 月 1 日开始所经过的秒数），由用户日志字段的 time 计算得到。
`__topic__`	日志的 topic。
`__source__`	日志来源的客户端 IP。

JSON格式存储会默认带上以上字段内容。

CSV存储可以根据您的需求自行选择。例如您需要日志的topic，那么可以填写字段名：`__topic__`。

OSS存储地址

压缩类型	文件后缀	OSS文件地址举例
无压缩	.csv	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.csv
snappy	.snappy.csv	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.snappy.csv

数据消费

HybridDB

建议配置如下：

- 分隔符 delimiter：逗号(,)
- 转义符 quote：双引号(“)
- 无效字段内容 null：不填写（空）
- 投递字段名称 header：不勾选（HybirdDB默认csv文件行首无字段说明）

其它

CSV是可读格式，可以直接从OSS下载以文本形式打开查看。

如果使用了snappy压缩，可以参考*JSON*格式的snappy解压缩说明。

12.2.3 RAM授权

配置OSS投递任务之前，OSS Bucket的拥有者需要配置快捷授权，授权完成后，当前账号的日志服务有权限对OSS Bucket进行写入操作。

本文为您介绍配置OSS投递任务中多种场景下的RAM授权操作。

- 如您需要对OSS Bucket进行更细粒度的访问控制，请参考修改授权策略。
- 如果日志服务Project和OSS Bucket不是同一阿里云账号创建的，请参考跨阿里云账号投递。
- 如果子账号需要将日志数据投递至其他阿里云账号下的OSS Bucket，请参考子账号的跨账号投递。
- 如果子账号需要将当前主账号的日志数据投递至同一账号下的OSS Bucket，请参考授权*RAM*用户。

修改授权策略

通过快捷授权，角色 AliyunLogDefaultRole 默认被授予 AliyunLogRolePolicy，具有账号B所有 OSS Bucket 的写入权限。

如您需要更精细的访问控制粒度，请解除角色 AliyunLogDefaultRole 的 AliyunLogRolePolicy 授权，并参考《*OSS*用户指南》创建更细粒度的权限策略，授权给角色 AliyunLogDefaultRole。

跨阿里云账号投递

如您的日志服务Project和OSS Bucket不是同一阿里云账号创建的，您需要按照以下步骤配置授权策略。

例如，需要将A账号下的日志服务数据投递至B账号创建的OSS Bucket中。

1. 主账号B通过快捷授权创建角色AliyunLogDefaultRole，并授予其OSS的写入权限。
2. 在访问控制RAM控制台中单击左侧导航栏角色管理，找到AliyunLogDefaultRole，并单击角色名称以查看该角色的基本信息。

   其中，该角色描述中`Service`配置项定义了角色的合法使用者，例如`log.aliyuncs.com`表示当前账号可扮演此角色以获取OSS的写入权限。
3. 在`Service`配置项中添加角色描述`A_ALIYUN_ID@log.aliyuncs.com`。账号A的账号ID请在**账号管理 > 安全设置**中查看。

   例如A 的主账号 ID 为1654218965343050，更改后的账号描述为：

```
{
"Statement": [
 {
   "Action": "sts:AssumeRole",
   "Effect": "Allow",
   "Principal": {
     "Service": [
       "1654218965343050@log.aliyuncs.com",
       "log.aliyuncs.com"
     ]
   }
 }
],
"Version": "1"
}
```

   该角色描述表示账号A有权限通过日志服务获取临时 Token 来操作 B 的资源，关于角色描述的更多信息请参考授权策略管理。
4. A账号创建投递任务，并在配置投递任务时，**RAM角色**一栏填写OSS Bucket拥有者的RAM角色标识ARN，即账号B通过快速授权创建的RAM角色AliyunLogDefaultRole。

   RAM角色的ARN可以在该角色的**基本信息**中查看，格式为`acs:ram::13234:role/logrole`。

子账号的跨账号投递

如果主账号A账号的子账号A1创建投递任务，将A账号的日志数据投递至B账号的OSS Bucket中，需要由主账号A为子账号A1授予PassRole权限。

配置步骤如下：

1. 参考跨阿里云账号投递，主账号B配置快捷授权，并添加角色描述。
2. 主账号A登录访问控制RAM控制台，为子账号A1授予AliyunRAMFullAccess权限。
   a. 主账号A在用户管理页面，单击子账号A1右侧的授权。

b. 在可选授权策略名称中查找**AliyunRAMFullAccess**，将其加入已选授权策略名称，并单击确定。

授权成功后，子账号A1将具备RAM所有权限。

如果您需要控制A1的权限范围，只授予投递 OSS 的必要权限，可以修改授权策略的`Action`和`Resource`参数。

示例授权策略如下，您需要将`Resource`的内容替换为AliyunLogDefaultRole的角色ARN。

```
{
"Statement": [
{
"Action": "ram:PassRole",
"Effect": "Allow",
"Resource": "acs:ram::1111111:role/aliyunlogdefaultrole"
}
],
"Version": "1"
}
```

c. 子账号A1创建投递任务，并在配置投递任务时，**RAM**角色一栏填写OSS Bucket拥有者的RAM角色标识ARN，即账号B通过快速授权创建的RAM角色AliyunLogDefaultRole。

12.3 投递日志到MaxCompute

12.3.1 通过DataWorks投递数据到MaxCompute

除了将日志投递到OSS存储之外，您还可以选择将日志数据通过DataWorks的数据集成（Data Integration）功能投递至MaxCompute。数据集成是阿里集团对外提供的稳定高效、弹性伸缩的数据同步平台，为阿里云大数据计算引擎（包括MaxCompute、AnalyticDB、OSPS）提供离线、批量的数据进出通道。

该功能支持的地域请参见*DataWorks*文档。

应用场景

- 跨Region的LogHub与MaxCompute等数据源的数据同步
- 不同阿里云账号下的LogHub与MaxCompute等数据源间的数据同步
- 同一阿里云账号下的LogHub与MaxCompute等数据源间的数据同步
- 公共云与金融云账号下的LogHub与MaxCompute等数据源间的数据同步

前提条件

1. 已开通日志服务、MaxCompute和DataWorks。

2. 已使用日志服务成功采集到日志数据，LogHub中已存在可投递的日志数据。
3. 数据源账号需要启用一对访问密钥Access Key。
4. 跨账号投递的情况下，需要先配置RAM授权。

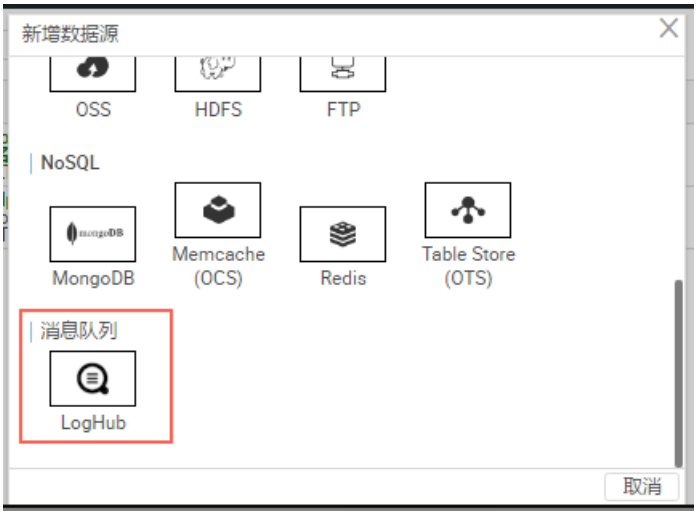
   详细配置请参考本文档中跨账号授权部分。

操作步骤

步骤1 创建数据源

1. 在DataWorks管理控制台打开数据集成页面，单击左侧导航栏的数据源页签。
2. 在数据源页面右上角单击新增数据源，弹出新增数据源页面。
3. 单击消息队列中的**LogHub**，进入新增**LogHub**数据源页面。

   图 12-4: 新增数据源



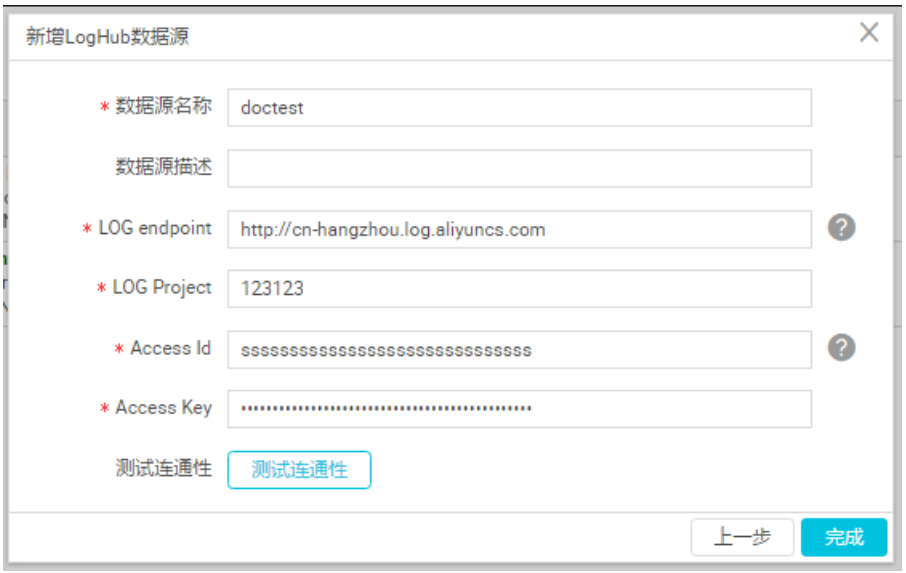
4. 填写数据源的配置项。

   配置项说明见下表。

配置项	说明
数据源名称	由英文字母、数字、下划线组成且须以字符或下划线开头，长度不超过60个字符。
数据源描述	对数据源进行简单描述，长度不超过80个字符。
LOG Endpoint	日志服务的Endpoint，根据您的Region确定，格式为：`http://yyy.com`。详细说明请参考服务入口。

配置项	说明
LOG Project	您想要投递至MaxCompute的日志服务Project。必须是已创建的Project。
Access Id/Access Key	数据源账号的访问密钥AccessKey（AK）相当于登录密码。您可以填写数据源主账号的AK或是子账号的AK，成功配置后，当前账号具备访问数据源账号日志数据的权限，可以将数据源账号的日志通过正在创建的同步任务进行投递。

图 12-5: 新增LogHub数据源



5. 单击测试连通性。页面右上角弹出提示测试连通性成功后，单击完成。

步骤2 配置同步任务

单击左侧导航的同步任务，并单击第二步新建同步任务，进入配置同步任务流程。

您可选择向导模式，通过简单便捷的可视化页面完成任务配置；或者选择脚本模式，深度自定义配置您的同步任务。

向导模式

共有选择来源、选择目标、字段映射、通道控制和预览保存五步操作。

1. 选择来源。

数据源请选择您在步骤1中配置的数据源。请参考下表填写所有配置项。

配置项	说明
数据源	选择来源的LogHub数据源的名称。
Logstore	导出增量数据的表的名称。该表需要开启Stream，可以在建表时开启，或者使用UpdateTable接口开启。
日志开始时间	数据消费的开始时间位点，为时间范围（左闭右开）的左边界，为yyyyMMddHHmmss格式的时间字符串（比如20180111013000），可以和DataWorks的调度时间参数配合使用。
日志结束时间	数据消费的结束时间位点，为时间范围（左闭右开）的右边界，为yyyyMMddHHmmss格式的时间字符串（比如20180111013010），可以和DataWorks的调度时间参数配合使用。
批量条数	一次读取的数据条数，默认为256。

填写完成后请单击**数据预览**的下拉按钮，展开**数据预览**详情。查看是否已拉取到日志数据，并单击下一步。

> **说明：**
> 数据预览是根据LogHub里的全部的数据选择几条展现在预览框，可能您同步的数据会跟您的预览的结果不一样，因为您同步的数据会指定开始时间和结束时间。

图 12-6: 选择来源



2. 选择目标。
   a. 选择MaxCompute数据源及目标表。

      如您还没有创建MaxCompute表，可以单击右侧的**一键生成目标表**。在弹出菜单中新建数据表。

   b. 填写分区信息。

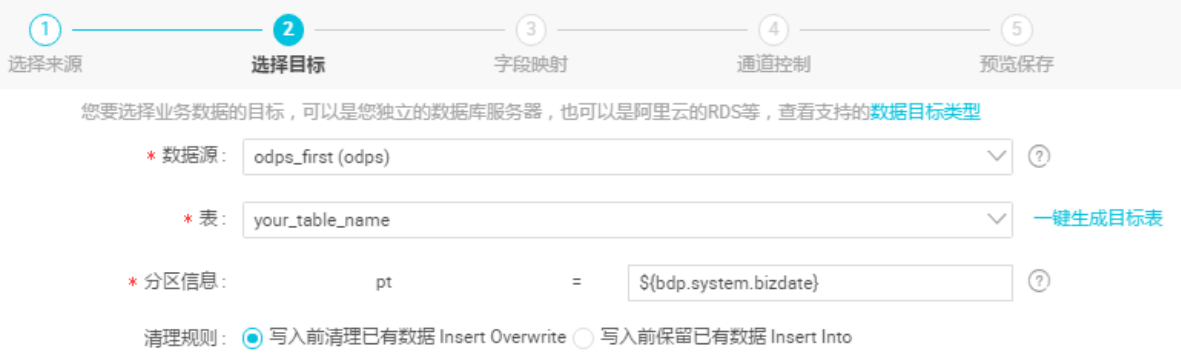
分区配置支持正则，比如分区pt值配置为*表示读取所有的pt分区数据。

c. 选择清理规则。

您可以选择在写入前清理已有数据，即覆盖式写数据，或者写入前保留已有数据，即插入式写数据。

完成后单击下一步。

图 12-7: 选择目标



3. 字段映射。

选择字段的映射关系。需对字段映射关系进行配置，左侧源头表字段和右侧目标表字段默认为一一对应的关系，您也可以单击右侧的同行映射以选择或取消同行映射。

说明：

- 如您需要将手动添加的日志字段作为同步列，请使用脚本模式配置。
- 可以输入常量，输入的值需要使用英文单引号包括，如’abc’、’123’等；
- 可以配合调度参数使用，如 ${bdp.system.bizdate} 等；
- 可以输入关系数据库支持的函数，如now()、count(1)等；
- 如果您输入的值无法解析，则类型显示为’未识别’。

图 12-8: 字段映射



4. 通道控制。

   配置作业速率上限和脏数据检查规则，如下图所示：

   图 **12-9:** 通道控制



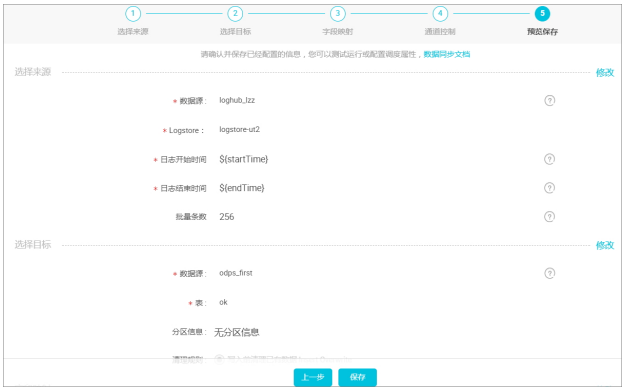
   配置项说明：

   - **DMU**：数据移动单位 (DMU) 是数据集成消耗资源（包含 CPU、内存、网络等资源分配）的度量单位。
   - **作业并发数**：可将此属性视为数据同步任务内，可从源并行读取或并行写入数据存储端的最大线程数。

5. 预览保存。

   完成以上配置后，上下滚动鼠标可查看任务配置，如若无误，单击**保存**。

   图 **12-10:** 预览保存



脚本模式

若您需要用脚本模式配置此任务，请参考如下脚本。

```
{
"type": "job",
"version": "1.0",
"configuration": {
"reader": {
"plugin": "loghub",
"parameter": {
"datasource": "loghub_lzz",//数据源名，保持跟您添加的数据源名一致
"logstore": "logstore-ut2",//目标日志库的名字，logstore是日志服务中日志数据的采集、存储和查询单元。
"beginDateTime": "${startTime}",//数据消费的开始时间位点，为时间范围（左闭右开）的左边界
"endDateTime": "${endTime}",//数据消费的结束时间位点，为时间范围（左闭右开）的右边界
"batchSize": 256,//一次读取的数据条数，默认为256。
"splitPk": "",
"column": [
"key1",
"key2",
"key3"
]
}
},
"writer": {
"plugin": "odps",
"parameter": {
"datasource": "odps_first",//数据源名，保持跟您添加的数据源名一致
"table": "ok",//目标表名
"truncate": true,
"partition": "",//分区信息
"column": [//目标列名
"key1",
"key2",
"key3"
]
}
},
"setting": {
"speed": {
"mbps": 8,/作业速率上限
"concurrent": 7//并发数
}
}
}
}
```

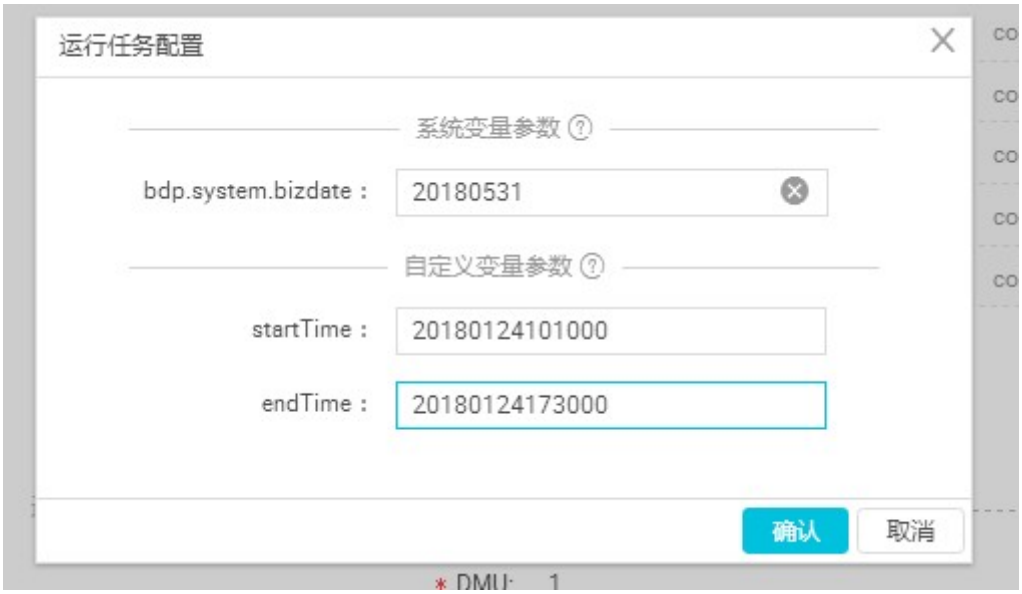
步骤3 运行任务

您可通过以下两种方式运行任务。

- 直接运行（一次性运行）

单击任务上方的运行按钮，将直接在数据集成页面运行任务。运行之前需要配置自定义参数的具体数值。

图 **12-11:** 运行任务配置



如上图所示，显示的时间是同步10：10到17:30这段时间的LogHub记录到MaxCompute。

- 调度运行

  单击提交按钮，将同步任务提交到调度系统中，调度系统会按照配置属性在从第二天开始自动定时执行。相关调度的配置请参见调度配置。

  设置5分钟调度一次，从00：00到23:59，startTime=$[yyyymmddhh24miss-10/24/60]系统前10分钟到 endTime=$[yyyymmddhh24miss-5/24/60]系统前5分钟时间。

  图 **12-12:** 调度配置



跨账号授权

配置跨账号的投递任务，需要通过访问控制RAM授权。

- 跨账号授权

  主账号之间的数据投递，可以直接在新增**LogHub**数据源步骤中，填入数据源主账号的AK。连通性测试通过后，表示授权成功。

  例如，A账号为日志数据源，将A账号收集的日志数据通过B账号开通的DataWorks服务投递至B账号的MaxCompute表中，需要B账号配置数据集成任务，并且在新增**LogHub**数据源步骤中，填入A账号的主账号AK。配置成功后，B账号有权限读取A账号下的所有日志数据。

- 子账号授权

  如果您不想暴露主账号的AK、或者需要投递子账号收集的日志数据，需要对子账号进行显式授权。

  — 赋予子账号管理权限

  如您需要通过子账号投递主账号名下的所有日志数据，需要按照以下步骤授权并配置AK。

  1. 主账号A为其名下的子账号A1赋予日志服务的管理权限，即`AliyunLogFullAccess`和`AliyunLogReadOnlyAccess`。详情请参见授权*RAM* 用户。
  2. B账号配置数据集成任务，并在新增**LogHub**数据源步骤中，填入数据源子账号的AK。

  完成以上步骤后，B账号有权限读取A账号下的所有日志数据。

  — 赋予子账号自定义权限

  如您需要通过子账号投递主账号名下的部分日志数据，需要按照以下步骤授权并配置AK。

  1. 主账号A为其子账号A1配置自定义授权策略。相关的授权请参见简介和概览。
  2. B账号配置数据集成任务，并在新增**LogHub**数据源步骤中，填入数据源子账号的AK。

  完成以上步骤后，B账号有权限读取A账号下的部分日志数据。

  自定义授权策略示例：

  此时，B账号通过子账号A1只能同步日志服务project_name1以及project_name2的数据。

```
{
"Version": "1",
"Statement": [
{
"Action": [
"log:Get*",
"log:List*",
"log:CreateConsumerGroup",
"log:UpdateConsumerGroup",
"log:DeleteConsumerGroup",
"log:ListConsumerGroup",
"log:ConsumerGroupUpdateCheckPoint",
```

```
"log:ConsumerGroupHeartBeat",
"log:GetConsumerGroupCheckPoint"
],
"Resource": [
"acs:log:*:*:project/project_name1",
"acs:log:*:*:project/project_name1/*",
"acs:log:*:*:project/project_name2",
"acs:log:*:*:project/project_name2/*"
],
"Effect": "Allow"
}
]
}
```

图 **12-13:** 自定义授权策略



12.3.2 通过日志服务投递日志到MaxCompute

投递日志到 MaxCompute 是日志服务的一个功能，能够帮助您最大化数据价值。您可以自己决定对某个日志库是否启用该功能。一旦启用该功能，日志服务后台会定时把写入到该日志库内的日志投递到 MaxCompute 对应的表格中。

使用限制

- 数加控制台创建、修改投递配置必须由主账号完成，不支持子账号操作。
- 投递MaxCompute是批量任务，请谨慎设置分区列及其类型：保证一个同步任务内处理的数据分区数小于512个；用作分区列的字段值不能为空或包括/等MaxCompute保留字段 。配置细节请参考下文投递配置说明。
- 不支持海外Region的MaxCompute投递，海外Region的MaxCompute请使用*dataworks*进行数据同步。国内Region投递支持如下：

日志服务Region	MaxCompute Region
华北1	华东2
华北2	华北2、华东2

日志服务Region	MaxCompute Region
华北3	华东2
华北5	华东2
华东1	华东2
华东2	华东2
华南1	华南1、华东2
香港	华东2

功能优势

日志服务收集的日志除了可以被实时查询外，还可以把日志数据投递到大数据计算服务MaxCompute（原ODPS），进一步进行个性化BI分析及数据挖掘。通过日志服务投递日志数据到MaxCompute具有如下优势：

- 使用便捷

  您只需要完成2步配置即可以把日志服务Logstore的日志数据迁移到MaxCompute中。

- 避免重复收集工作

  由于日志服务的日志收集过程已经完成不同机器上的日志集中化，无需重复在不同机器上收集一遍日志数据后再导入到MaxCompute。

- 充分复用日志服务内的日志分类管理工作

  用户可让日志服务中不同类型的日志（存在不同Logstore中）、不同Project的日志自动投递到不同的MaxCompute表格，方便管理及分析MaxCompute内的日志数据。

 说明：

一般情况下日志数据在写入Logstore后的1个小时导入到MaxCompute，您可以在控制台投递任务管理查看导入状态。导入成功后即可在MaxCompute内查看到相关日志数据。判断数据是否已完全投递请参考文档。

结合日志服务的实时消费，投递日志数据到MaxCompute的数据通道以及日志索引功能，可以让用户按照不同的场景和需求、以不同的方式复用数据，充分发挥日志数据的价值。

配置流程

举例日志服务的一条日志如下：

```
16年01月27日20时50分13秒
10.10.*.*
ip:10.10.*.*
status:200
thread:414579208
time:27/Jan/2016:20:50:13 +0800
url:POST /PutData?Category=YunOsAccountOpLog&AccessKeyId
=****************&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&
Topic=raw&Signature=******************************** HTTP/1.1
user-agent:aliyun-sdk-java
```

日志左侧的ip、status、thread、time、url、user-agent等是日志服务数据的字段名称，需要在下方配置中应用到。

步骤1 初始化数加平台

1. 在日志服务的控制台Logstore列表单击日志投递列的MaxCompute。

   自动跳转到初始化数加平台的页面。MaxCompute默认为按量付费模式，具体参见MaxCompute文档说明。

2. 查看服务协议和条款后单击确定，初始化数加平台。

   初始化开通需10~20秒左右，请耐心等待。如果已经开通数加及大数据计算服务MaxCompute（原ODPS），将直接跳过该步骤。

步骤2 数据模型映射

在日志服务和大数据计算服务MaxCompute（原ODPS）之间同步数据，涉及两个服务的数据模型映射问题。您可以参考日志服务日志数据结构了解数据结构。

将样例日志导入MaxCompute，分别定义MaxCompute数据列、分区列与日志服务字段的映射关系：

MaxCompu 列类型	MaxCompute 列名（可自定义）	MaxCompu 列类型（可自定义）	日志服务字段名（投递配置里填写）	日志服务字段类型	日志服务字段语义
数据列	log_source	string	__source__	系统保留字段	日志来源的机器 IP。

MaxCompu 列类型	MaxCompute 列名（可自定义）	MaxCompu 列类型（可自定义）	日志服务字段名（投递配置里填写）	日志服务字段类型	日志服务字段语义
	log_time	bigint	__time__	系统保留字段	日志的 Unix 时间戳（是从 1970 年 1 月 1 日开始所经过的秒数），对应数据模型中的Time域。
	log_topic	string	__topic__	系统保留字段	日志主题。
	time	string	time	日志内容字段	解析自日志，对应数据模型中的key-value，例如Logtail采集的数据在很多时候__time__与time取值相同。
	ip	string	ip	日志内容字段	解析自日志。
	thread	string	thread	日志内容字段	解析自日志。
	log_extract_others	string	__extract_others__	系统保留字段	未在配置中进行映射的其他日志内字段会通过 key-value 序列化到json，该 json 是一层结构，不支持字段内部 json 嵌套。
分区列	log_partition_time	string	__partition_time__	系统保留字段	由日志的 __time__ 字段对齐计算而得，分区粒度可配置，在配置项部分详述。
	status	string	status	日志内容字段	解析自日志，该字段取值应该是可以枚举的，保证分区数目不会超出上限。

- MaxCompute 表至少包含一个数据列、一个分区列。
- 系统保留字段中建议使用 __partition_time__，__source__，__topic__。

- MaxCompute 单表有分区数目 6 万的限制，分区数超出后无法再写入数据，所以日志服务导入MaxCompute表至多支持3个分区列。请谨慎选择自定义字段作为分区列，保证其值是可枚举的。
- 系统保留字段 __extract_others__ 历史上曾用名 _extract_others_，填写后者也是兼容的。
- MaxCompute 分区列的值不支持”/“等特殊字符，这些是 MaxCompute 的保留字段。
- MaxCompute 分区列取值不支持空，所以映射到分区列的字段必须出自保留字段或日志字段，且可以通过cast运算符将string类型字段值转换为对应分区列类型，空分区列的日志会在投递中被丢弃。
- 日志服务数据的一个字段最多允许映射到一个MaxCompute表的列（数据列或分区列），不支持字段冗余，同一个字段名第二次使用时其投递的值为null，如果null出现在分区列会导致数据无法被投递。

步骤3 配置投递规则

1. 开启投递。

   初始化数加平台之后，根据页面提示进入**LogHub** —— 数据投递页面，选择需要投递的Logstore，并单击开启投递。

   您也可以在MaxCompute（原ODPS）投递管理页面选择需要投递的Logstore，并单击开启投递以进入**LogHub** —— 数据投递页面。

   图 **12-14:** 开启投递



2. 配置投递规则。

   在**LogHub** —— 数据投递页面配置字段关联等相关内容。

图 12-15: 配置投递规则



配置项含义：

参数	语义
投递名称	自定义一个投递的名称，方便后续管理。
MaxCompute Project	MaxCompute项目名称，该项默认为新创建的Project，如果已经是MaxCompute老客户，可以下拉选择已创建其他Project。
MaxCompute Table	MaxCompute表名称，请输入自定义的新建的MaxCompute表名称或者选择已有的MaxCompute表。
MaxCompute 普通列	按序，左边填写与MaxCompute表数据列相映射的日志服务字段名称，右边填写或选择MaxCompute表的普通字段名称及字段类型。
MaxCompute 分区列	按序，左边填写与MaxCompute表分区列相映射的日志服务字段名称，右边填写或选择MaxCompute表的普通字段名称及字段类型。
分区时间格式	__partition_time__输出的日期格式，参考 *Java SimpleDateFormat* 。
导入MaxCompute间隔	MaxCompute数据投递间隔，默认1800，单位：秒。

- 该步会默认为客户创建好新的MaxCompute Project和Table，其中如果已经是MaxCompute老客户，可以下拉选择其他已创建Project。
- 日志服务投递MaxCompute功能按照字段与列的顺序进行映射，修改MaxCompute表列名不影响数据导入，如更改MaxCompute表schema，请重新配置字段与列映射关系。

参考信息

__partition_time__ 格式

将日志时间作为分区字段，通过日期来筛选数据是MaxCompute常见的过滤数据方法。

`__partition_time__` 是根据日志__time__值计算得到（不是日志写入服务端时间，也不是日志投递时间），结合分区时间格式，向下取整（为避免触发MaxCompute单表分区数目的限制，日期分区列的值会按照导入MaxCompute间隔对齐）计算出日期作为分区列。

举例来说，日志提取的time字段是“27/Jan/2016:20:50:13 +0800”，日志服务据此计算出保留字段__time__为1453899013（Unix时间戳），不同配置下的时间分区列取值如下：

导入MaxCompute间隔	分区时间格式	__partition_time__
1800	yyyy_MM_dd_HH_mm_00	2016_01_27_20_30_00
1800	yyyy-MM-dd HH:mm	2016-01-27 20:30
1800	yyyyMMdd	20160127

导入MaxCompute间隔	分区时间格式	__partition_time__
3600	yyyyMMddHHmm	201601272000
3600	yyyy_MM_dd_HH	2016_01_27_20

- 请勿使用精确到秒的日期格式：1. 很容易导致单表的分区数目超过限制（6万）；2. 单次投递任务的数据分区数目必须在512以内。
- 以上分区时间格式是测试通过的样例，您也可以参考*Java SimpleDateFormat*自己定义日期格式，但是该格式不得包含斜线字符”/“（这是MaxCompute的保留字段）。

__partition_time__ 使用方法

使用MaxCompute的字符串比较筛选数据，可以避免全表扫描。比如查询2016年1月26日一天内日志数据：

```
select * from {ODPS_TABLE_NAME} where log_partition_time >= "
2015_01_26" and log_partition_time < "2016_01_27";
```

__extract_others__使用方法

log_extract_others为一个json字符串，如果想获取该字段的user-agent内容，可以进行如下查询：

```
select get_json_object(sls_extract_others, "$.user-agent") from {
ODPS_TABLE_NAME} limit 10;
```



说明：

- get_json_object是*MaxCompute*提供的标准*UDF*。请联系MaxCompute团队开通使用该标准UDF的权限。
- 示例供参考，请以MaxCompute产品建议为最终标准。

其他操作

编辑投递配置

在Logstore列表投递项，单击“修改”即可针对之前的配置信息进行编辑。其中如果想新增列，可以在大数据计算服务MaxCompute（原ODPS）修改投递的数据表列信息，则点击“修改”后会加载最新的数据表信息。

投递任务管理

在启动投递功能后，日志服务后台会定期启动离线投递任务。用户可以在控制台上看到这些投递任务的状态和错误信息。具体请参考管理日志投递任务。

如果投递任务出现错误，控制台上会显示相应的错误信息：

错误信息	建议方案
MaxCompute项目空间不存在	在MaxCompute控制台中确认配置的MaxCompute项目是否存在，如果不存在则需要重新创建或配置。
MaxCompute表不存在	在MaxCompute控制台中确认配置的MaxCompute表是否存在，如果不存在则需要重新创建或配置。
MaxCompute项目空间或表没有向日志服务授权	在MaxCompute控制台中确认授权给日志服务账号的权限是否还存在，如果不存在则需要重新添加上相应权限。
MaxCompute错误	显示投递任务收到的MaxCompute错误，请参考MaxCompute相关文档或联系MaxCompute团队解决。日志服务会自动重试最近两天时间的失败任务。
日志服务导入字段配置无法匹配MaxCompute表的列	重新配置MaxCompute表格的列与日志服务数据字段的映射配置。

当投递任务发生错误时，请查看错误信息，问题解决后可以通过云控制台中“日志投递任务管理”或SDK来重试失败任务。

MaxCompute中消费日志

MaxCompute用户表中示例数据如下：

```
| log_source | log_time | log_topic | time | ip | thread | log_extrac
t_others | log_partition_time | status |
+------------+------------+-----------+-----------+-----------
+-----------+-----------------+--------------------+-----------+
| 10.10.*.* | 1453899013 | | 27/Jan/2016:20:50:13 +0800 | 10.10.*.*
 | 414579208 | {"url":"POST /PutData?Category=YunOsAccountOpLog&
AccessKeyId=***************&Date=Fri%2C%2028%20Jun%202013%2006%3A53%
3A30%20GMT&Topic=raw&Signature=****************************** HTTP/1
.1","user-agent":"aliyun-sdk-java"} | 2016_01_27_20_50 | 200 |
+------------+------------+-----------+-----------+-----------+-----------
+-----------+-----------------+--------------------+-----------+
```

同时，我们推荐您直接使用已经与MaxCompute绑定的大数据开发Data IDE来进行可视化的BI分析及数据挖掘，这将提高数据加工的效率。

授予MaxCompute数据投递权限

如果在数加平台执行表删除重建动作，会导致默认授权失效。请手动重新为日志服务投递数据授权。

在MaxCompute项目空间下添加用户：

```
ADD USER aliyun$shennong_open@aliyun.com;
```

shennong_open@aliyun.com 是日志服务系统账号（请不要用自己的账号），授权目的是为了能将数据写入到MaxCompute

MaxCompute项目空间Read/List权限授予：

```
GRANT Read, List ON PROJECT {ODPS_PROJECT_NAME} TO USER aliyun$
shennong_open@aliyun.com;
```

MaxCompute项目空间的表Describe/Alter/Update权限授予：

```
GRANT Describe, Alter, Update ON TABLE {ODPS_TABLE_NAME} TO USER
aliyun$shennong_open@aliyun.com;
```

确认MaxCompute授权是否成功：

```
SHOW GRANTS FOR aliyun$shennong_open@aliyun.com;

A       projects/{ODPS_PROJECT_NAME}: List | Read
A       projects/{ODPS_PROJECT_NAME}/tables/{ODPS_TABLE_NAME}:
Describe | Alter | Update
```

12.4 管理日志投递任务

投递日志是日志服务的一个功能，能够帮助您最大化数据价值。您可以选择将收集到的日志数据通过控制台方式投递至MaxCompute，做数据长期存储或联合其它系统（如 E-MapReduce）消费数据。一旦启用日志投递功能，日志服务后台会定时把写入到该日志库内的日志投递到对应云产品中。为方便您及时了解投递进度，处理线上问题，日志服务控制台提供了 **日志投递任务管理** 页面，您可以查询指定时间内的数据投递状态。

在**Logstore列表**界面，单击**日志投递**列下的或**MaxComputeOSS**，进入**日志投递任务管理**页面。您可以执行以下操作管理您的投递任务：

开启/关闭投递任务

1. **日志投递任务管理**页面左上角菜单中选择目标Logstore。
2. 单击Logstore菜单旁的**开启投递**或**关闭投递**。

   关闭投递任务后再次开启任务，需要重新配置投递规则。

关闭投递任务后再次开启任务，需要重新配置投递规则。

修改投递规则

成功创建投递任务之后，可以单击**修改属性**修改投递规则。

查看投递任务详情

根据Logstore、时间段和任务投递状态筛选出需要查看的投递任务后，可以在当前页面中查看指定投递任务状态、开始时间、结束时间、接受日志数据时间、任务类型等详细信息。

任务的投递状态有以下三种：

状态	含义	操作
成功	投递任务正常运行状态。	无须关注。
进行中	投递任务进行中。	请稍后查看是否投递成功。
失败	日志数据投递失败。投递任务因外部原因出现了错误，且无法重试。如MaxCompute表结构不符合日志服务规范、无授权等。	请排查投递问题。

删除投递配置

操作步骤

1. 在**Logstore列表**页面，单击**删除规则**。
2. 在弹出的对话框中单击**确定**。

   删除后将不能再创建同名的离线归档配置，请慎重选择。

13 访问控制 RAM

13.1 简介

RAM (Resource Access Management) 是阿里云为客户提供的 用户身份管理 与 资源访问控制 服务。使用 RAM，您可以创建、管理用户账号（比如员工、系统或应用程序），并可以控制这些用户账号对您名下资源具有的操作权限。当您的企业存在多用户协同操作资源时，使用 RAM 可以让您避免与其他用户共享云账号密钥，按需为用户分配最小权限，从而降低您的企业信息安全风险。

为了更精细地管理和操作日志服务资源，您可以通过阿里云RAM产品为您名下的子账号、日志服务的RAM服务角色和用户角色赋予相应的访问权限。

身份管理

您可以通过RAM进行用户身份管理。例如在您的账号下创建并管理用户账号/用户组、创建服务角色以代表日志服务、创建用户角色以进行跨账号的资源操作与授权管理。

日志服务支持收集API网关、SLB等云产品的日志数据，您需要在配置前通过快速授权页完成服务角色的创建与授权。

角色	默认权限	说明
AliyunLogArchiveRole	AliyunLogArchiveRolePolicy	日志服务默认使用此角色访问您的SLB云产品日志，默认授权策略用于导出SLB服务日志。快速授权请单击快速授权页
AliyunLogDefaultRole	AliyunLogRolePolicy	用于日志服务默认角色的授权策略，包含OSS的写入权限。快速授权请单击快速授权页。
AliyunLogETLRole	AliyunLogETLRolePolicy	用于日志服务ETL功能角色的授权策略，日志服务默认使用此角色来访问您在其他云产品中的资源。快速授权请单击快速授权页。
AliyunMNSLoggingRole	AliyunMNSLoggingRolePolicy	日志服务默认使用此角色访问您的MNS云产品日志，默认授权策略用于导出MNS服务日

角色	默认权限	说明
		志，包含OSS的写入权限。快速授权请单击快速授权页。

资源访问控制

您可以为名下的用户账号/用户组以及角色授予对应的授权策略。

您也可以创建自定义授权策略，或者以自定义授权策略和系统授权策略为模板，参考概览编辑更细粒度的授权策略。

日志服务支持以下系统授权策略。

授权策略	类型	说明
AliyunLogFullAccess	系统策略	日志服务的全部管理权限。
AliyunLogReadOnlyAccess	系统策略	只读访问日志服务的权限。

应用场景

授权RAM子用户访问日志服务

在实际的应用场景中，主账号可能需要将日志服务的运营维护工作交予其名下的RAM子用户，由RAM子用户对日志服务进行日常维护工作；或者主账号名下的RAM子用户可能有访问日志服务资源的需求。此时，主账号需要对其名下的RAM子用户进行授权，授予其访问或者操作日志服务的权限。出于安全性的考虑，日志服务建议您将RAM子用户的权限设置为需求范围内的最小权限。

配置详情请参考授权RAM 用户。

授权服务角色读日志

日志服务目前提供基于用户日志内容报警功能，为了读取日志数据，需要用户显式授权日志服务服务账号访问用户数据。

配置详情请参考授权服务角色。

授权用户角色操作日志服务

RAM 用户角色是一种虚拟用户，它没有确定的身份认证密钥，且需要被一个受信的实体用户（比如云账号、RAM-User 账号、云服务账号）扮演才能正常使用。扮演成功后实体用户将获得 RAM 用户角色的临时安全令牌，使用这个临时安全令牌就能以RAM用户角色身份访问被授权的资源。

- 将日志服务的操作权限授予一个受信实体用户，允许该实体用户下的RAM角色操作日志服务。配置详情请参考授权服务角色。
- 授权移动应用客户端通过直连方式访问日志服务，将APP的日志直接上传到日志服务中。配置详情请参考授权移动应用直连日志服务。

13.2 授权RAM 用户

背景信息

在实际的应用场景中，主账号可能需要将日志服务的运营维护工作交予其名下的RAM用户，由RAM用户对日志服务进行日常维护工作；或者主账号名下的RAM用户可能有访问日志服务资源的需求。此时，主账号需要对其名下的RAM用户进行授权，授予其访问或者操作日志服务的权限。出于安全性的考虑，日志服务建议您将RAM用户的权限设置为需求范围内的最小权限。

主账号授权RAM用户访问日志服务资源，需要按照以下步骤完成。关于RAM用户的详细信息，请参考简介。

操作步骤

1. 创建子用户。
   a) 登录访问控制服务控制台。
   b) 在用户管理页面单击页面右上角的新建用户。
   c) 填写用户信息，勾选 为该用户自动生成**AccessKey** 并单击 确定。
   d) 在弹出的验证对话框中，单击 点击获取 并输入手机验证码，然后单击 确定。您可以在用户列表中看到创建的用户。
2. 授权子用户访问日志服务资源。

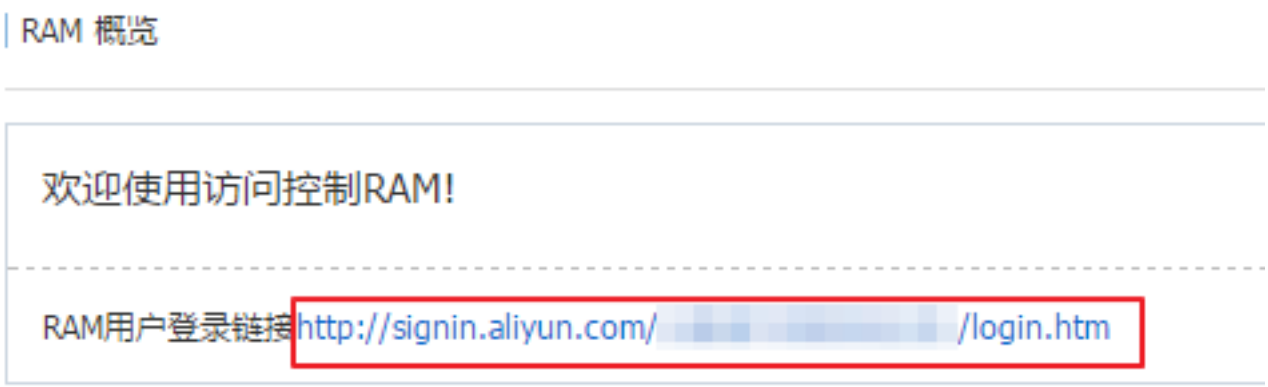
   日志服务提供两种系统授权策略，即`AliyunLogFullAccess`和`AliyunLogReadOnlyAccess`，分别表示管理权限和只读权限。您还可以在RAM控制台自定义授权策略，创建方法参考创建自定义授权策略，权限策略示例请参考*RAM*自定义授权场景和日志服务*RAM*授权策略。本文档以赋予用户`AliyunLogReadOnlyAccess`权限为例。

   a) 在 用户管理 页面找到对应的子用户，单击 授权。
   b) 在弹出的对话框中，选择 **AliyunLogReadOnlyAccess**。
   c) 在弹出的验证对话框中，单击 点击获取 并输入手机验证码，然后单击 确定。
3. RAM登录控制台。

完成创建用户和用户授权之后，用户就有权限访问日志服务控制台了。您可以通过以下两种方式以RAM用户身份登录控制台。

a) 在访问控制服务控制台概览页面，单击**RAM用户登录链接**，使用步骤1中创建的RAM用户用户名和密码登录。

图 **13-1: RAM**用户登录



b) 直接访问*RAM*用户通用登录页面，使用步骤1中创建的RAM用户用户名和密码登录。

其中的 **[企业号别名]** 默认为主账号的账号id（ali uid），可以在RAM控制台的设置 > 账户别名设置中查看并设置您的云帐号别名。

13.3 RAM自定义授权场景

通过RAM访问控制可以为名下的RAM用户（子用户）授权。

主账号可以对其名下的RAM用户（子用户）进行授权，授予其访问或者操作日志服务的权限。您可以为RAM用户授予系统授权策略和自定义授权策略。

注意事项

- 出于安全性的考虑，日志服务建议您将RAM用户的权限设置为需求范围内的最小权限。
- 通常情况下，您需要为RAM用户授予Project列表的只读权限，否则RAM用户无法进入Project列表查看资源。
- 动作`log:ListProject`提供Project列表的查看权限：
  - 具备此权限时，支持只读查看所有Project，暂不支持仅查看某几个Project。
  - 不具备此权限时，无法查看任何Project。

本文档为您演示常见的自定义授权场景和授权内容，包括：

- 控制台场景*#Project*列表和指定*Project*的只读权限

- 控制台场景#指定*Logstore*的只读权限和快速查询的创建、使用权限
- 控制台场景#指定*Project*中所有快速查询、仪表盘和指定*Logstore*的只读权限
- *API*场景#指定*Project*的写入权限
- *API*场景#指定*Project*的消费权限
- *API*场景#指定*Logstore*的消费权限

更多信息：

- 可授权的资源
- 可授权的动作
- 鉴权规则

控制台场景：**Project**列表和指定**Project**的只读权限

例如，主账号需要赋予RAM用户以下权限：

1. RAM用户可以看到主账号的日志服务Project列表。
2. RAM用户对主账号的指定Project有只读的访问权限。

同时满足1、2的授权策略如下：

```
{
   "Version": "1",
   "Statement": [
     {
       "Action": ["log:ListProject"],
       "Resource": ["acs:log:*:*:project/*"],
       "Effect": "Alow"
      },
     {
       "Action": [
         "log:Get*",
         "log:List*"
       ],
       "Resource": "acs:log:*:*:project/<指定的project名称>/*",
       "Effect": "Allow"
     }
   ]
 }
```

控制台场景：指定**Logstore**的只读权限和快速查询的创建、使用权限

例如，主账号需要赋予RAM用户以下权限：

1. RAM用户登录控制台可以看到主账号的日志服务Project列表。
2. RAM用户对指定Logstore具有只读权限，且可以创建、使用快速查询。

同时满足1、2的授权策略如下：

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:ListProject"
      ],
      "Resource": "acs:log:*:*:project/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:List*"
      ],
      "Resource": "acs:log:*:*:project/<指定的Project名称>/logstore/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:Get*",
        "log:List*"
      ],
      "Resource": [
        "acs:log:*:*:project/<指定的Project名称>/logstore/<指定的Logstore
名称>"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:List*"
      ],
      "Resource": [
        "acs:log:*:*:project/<指定的Project名称>/dashboard",
        "acs:log:*:*:project/<指定的Project名称>/dashboard/*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:Get*",
        "log:List*",
        "log:Create*"
      ],
      "Resource": [
        "acs:log:*:*:project/<指定的Project名称>/savedsearch",
        "acs:log:*:*:project/<指定的Project名称>/savedsearch/*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

控制台场景：指定Project中所有快速查询、仪表盘和指定Logstore的只读权限

例如，主账号需要赋予RAM用户以下权限：

1. RAM用户可以看到主账号的日志服务Project列表。
2. RAM用户仅能查看指定Logstore，同时可以查看所有的快速查询和仪表盘列表。



说明：

如果希望赋予RAM用户指定Logstore的只读权限，则必须同时赋予该RAM用户所有的快速查询和仪表盘列表的查看权限。

同时满足1、2的授权策略如下：

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:ListProject"
      ],
      "Resource": "acs:log:*:*:project/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:List*"
      ],
      "Resource": "acs:log:*:*:project/<指定的Project名称>/logstore/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:Get*",
        "log:List*"
      ],
      "Resource": [
        "acs:log:*:*:project/<指定的Project名称>/logstore/<指定的Logstore
名称>"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:Get*",
        "log:List*"
      ],
      "Resource": [
        "acs:log:*:*:project/<指定的Project名称>/dashboard",
        "acs:log:*:*:project/<指定的Project名称>/dashboard/*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:Get*",
        "log:List*"
      ],
      "Resource": [
        "acs:log:*:*:project/<指定的Project名称>/savedsearch",
```

```
        "acs:log:*:*:project/<指定的Project名称>/savedsearch/*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

API场景：指定Project的写入权限

RAM用户只能向某一Project写入数据，无法进行查询等其他操作。

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:Post*"
      ],
      "Resource": "acs:log:*:*:project/<指定的project名称>/*",
      "Effect": "Allow"
    }
  ]
}
```

API场景：指定Project的消费权限

RAM用户只能消费某一Project的数据，无法进行数据写入、查询等其他操作。

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:GetCursorOrData",
        "log:GetConsumerGroupCheckPoint",
        "log:UpdateConsumerGroup",
        "log:ConsumerGroupHeartBeat",
        "log:ConsumerGroupUpdateCheckPoint",
        "log:ListConsumerGroup",
        "log:CreateConsumerGroup"
      ],
      "Resource": "acs:log:*:*:project/<指定的project名称>/*",
      "Effect": "Allow"
    }
  ]
}
```

API场景：指定Logstore的消费权限

RAM用户只能消费指定Logstore的数据，无法进行数据写入、查询等其他操作。

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:GetCursorOrData",
```

```
        "log:GetConsumerGroupCheckPoint",
        "log:UpdateConsumerGroup",
        "log:ConsumerGroupHeartBeat",
        "log:ConsumerGroupUpdateCheckPoint",
        "log:ListConsumerGroup",
        "log:CreateConsumerGroup"
      ],
      "Resource": [
        "acs:log:*:*:project/<指定的project名称>/logstore/<指定的Logstore
名称>",
        "acs:log:*:*:project/<指定的project名称>/logstore/<指定的Logstore
名称>/*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

13.4 授权服务角色

日志服务目前提供基于用户日志内容报警功能，为了读取日志数据，需要用户显式授权日志服务服务账号访问用户数据。如果已经阅读过该文档完成授权，可以略过以下内容直接创建报警规则，具体授权步骤如下具体说明。

创建访问控制（RAM）角色

1. 用户登录访问控制控制台，打开角色管理功能，单击右上角新建角色按钮，第一步角色类型选择服务角色。

   图 13-2: 选择角色类型



2. 类型信息选择LOG日志服务。

图 **13-3:** 填写类型信息



3. 填写角色名称为**aliyunlogreadrole**（默认会扮演该角色访问数据，请不要修改名称），单击创建。

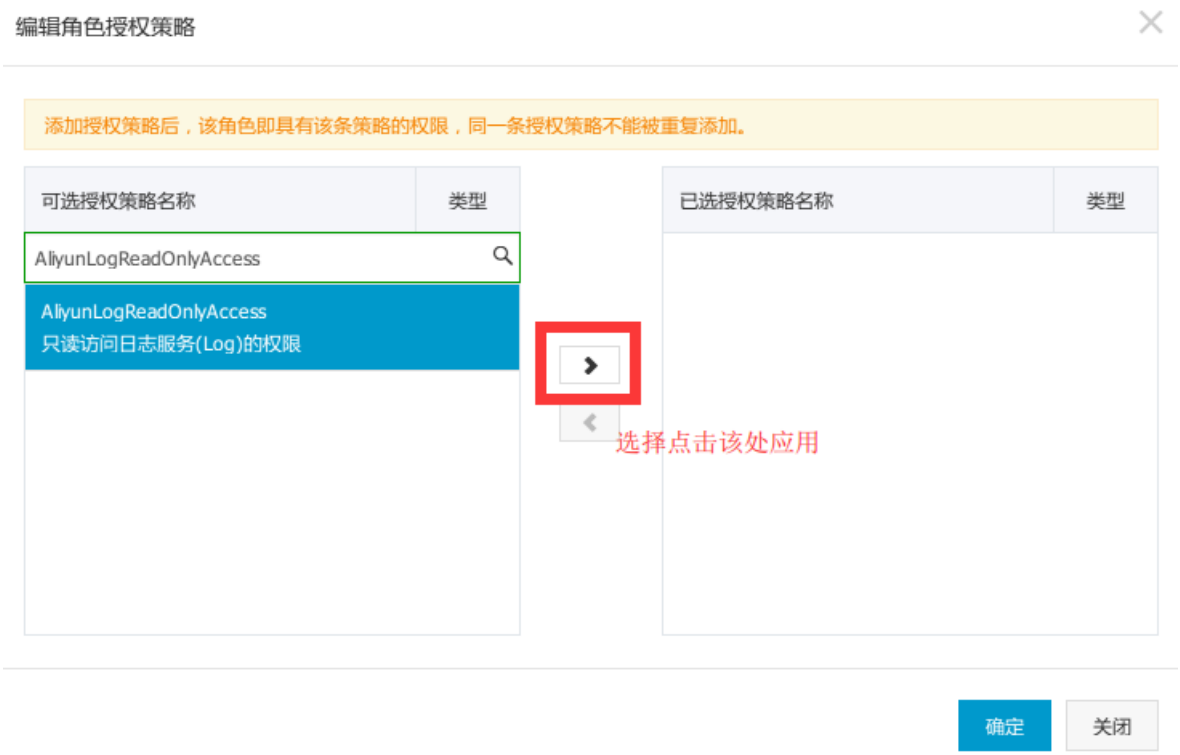
图 **13-4:** 配置角色基本信息



授权角色访问日志数据权限

角色创建完成后，在角色管理列表中，找到**aliyunlogreadrole**角色名称，点击授权，搜索**AliyunLogReadOnlyAccess**权限，直接应用。

图 **13-5:** 编辑角色授权策略



完成如上步骤后，日志服务即有权限定期读取指定日志库数据进行报警检查。

13.5 授权用户角色

角色，与用户一样，都是 RAM 中使用的身份。与 RAM 用户相比，RAM 用户角色是一种虚拟用户，它没有确定的身份认证密钥，且需要被一个受信的实体用户（比如云账号、RAM-User 账号、云服务账号）扮演才能正常使用。扮演成功后实体用户将获得 RAM 用户角色的临时安全令牌，使用这个临时安全令牌就能以RAM用户角色身份访问被授权的资源。

如果您需要将日志服务的操作权限授予一个受信实体用户，允许该实体用户下的RAM角色操作日志服务，您需要创建RAM用户角色并指定受信云账号、为RAM用户角色授权、为受信账号下的RAM用户授予AssumeRole权限、获取RAM用户角色的临时安全令牌。

更多内容请参考用户。

步骤1 创建用户角色并指定受信云账号

1. 登录到RAM控制台，并在左侧导航栏单击 角色管理。
2. 单击右上角新建角色。
3. 在选择角色类型子页，选择用户角色。
4. 在填写类型信息子页，选择受信云账号。

> 说明：
>
> - 若创建的角色是给您自己名下的 RAM 用户使用（比如授权移动 App 客户端直接操作LOG资源），请选择当前云账号为受信云账号。
> - 若创建的角色是给其他云账号名下的 RAM 用户使用（比如跨账号的资源授权），请选择 其他云账号，并在受信云账号 ID 中填写其他云账号的 ID。

图 13-6: 创建角色



5. 在配置角色基本信息子页，输入角色名称和备注后，单击创建。

步骤2 为RAM用户角色授权

成功创建用户角色后，该用户角色没有任何权限，您需要为RAM用户角色授予操作日志服务的权限。您上一步中指定的受信云账号将有权限扮演该RAM用户角色操作日志服务。

> 说明：

您可以赋予RAM用户角色一个或多个授权策略，包括系统授权策略和自定义授权策略。本文档以授予RAM用户角色管理日志服务的权限为例。

1. 在RAM控制台左侧导航栏单击角色管理。
2. 单击目标RAM用户角色名称右侧的授权。
3. 选择**AliyunLogFullAccess**权限，并单击确定。

更多内容请参考授权。

步骤3 为受信云账号的**RAM**用户授权

RAM用户角色需要被一个受信的实体用户扮演才能正常使用，但是受信实体用户不能以自己的身份扮演RAM用户角色，必须以RAM用户的身份和形式扮演。即**RAM** 用户角色只能通过 **RAM** 用户身份来扮演使用。

另外，受信云账号必须为其名下的RAM用户进行**AssumeRole**授权，授予该RAM用户调用STS服务AssumeRole接口的权限，此RAM用户才能代表受信云账号扮演步骤1中创建的RAM用户角色。

1. 登录受信云账号的RAM控制台。
2. 在用户管理页面中，选定用于授权的RAM用户，并单击右侧的授权。

   如果您之前没有创建过RAM用户，请参考用户创建一个RAM用户。
3. 选择系统授权策略**AliyunSTSAssumeRoleAccess**并单击确定。
4. 单击点击获取以获取验证码，输入收到的验证码，单击确认。

步骤4 获取**RAM**用户角色的临时安全令牌

当 RAM 用户被授予 AssumeRole 权限之后，可以使用其 AccessKey 调用安全令牌服务(STS) 的 AssumeRole 接口，以获取某个角色的临时安全令牌。

关于 AssumeRole API 的调用方法，请参考使用入门。

使用STS SDK拿到AccessKeyId、AccessKeySecret 、SecurityToken 之后就可以使用日志服务的SDK访问日志服务了。

下面是使用AccessKeyId、AccessKeySecret 、SecurityToken初始化LogClient的示例，Java SDK使用请参考*Java SDK*。

```
package sdksample;
import java.util.ArrayList;
import java.util.List;
import java.util.Vector;
import java.util.Date;
import com.aliyun.openservices.log.Client;
```

```
import com.aliyun.openservices.log.common.*;
import com.aliyun.openservices.log.exception.*;
import com.aliyun.openservices.log.request.*;
import com.aliyun.openservices.log.response.*;
import com.aliyun.openservices.log.common.LogGroupData;
import com.aliyun.openservices.log.common.LogItem;
import com.aliyun.openservices.log.common.Logs.Log;
import com.aliyun.openservices.log.common.Logs.Log.Content;
import com.aliyun.openservices.log.common.Logs.LogGroup;
import com.aliyun.openservices.log.common.Consts.CursorMode;
public class sdksample {
    public static void main(String args[]) throws LogException,
InterruptedException {
        String endpoint = "<log_service_endpoint>"; // 选择与上面步骤创
建 Project 所属区域匹配的Endpoint
        String accessKeyId = "<your_access_key_id>"; // 使用您的阿里云访
问密钥 AccessKeyId
        String accessKeySecret = "<your_access_key_secret>"; // 使用您
的阿里云访问密钥AccessKeySecret
    String securityToken = "<your_security_token>"; //角色的SecurityTo
ken
        String project = "<project_name>"; // 上面步骤创建的项目名称
        String logstore = "<logstore_name>"; // 上面步骤创建的日志库名称
        // 构建一个客户端实例
        Client client = new Client(endpoint, accessKeyId, accessKeyS
ecret);
    // 设置SecurityToken
    client.SetSecurityToken(securityToken);
        // 写入日志
        String topic = "";
        String source = "";
        // 连续发送 10 个数据包，每个数据包有 10 条日志
        for (int i = 0; i < 10; i++) {
            Vector<LogItem> logGroup = new Vector<LogItem>();
            for (int j = 0; j < 10; j++) {
                LogItem logItem = new LogItem((int) (new Date().
getTime() / 1000));
                logItem.PushBack("index"+String.valueOf(j), String.
valueOf(i * 10 + j));
                logGroup.add(logItem);
            }
            PutLogsRequest req2 = new PutLogsRequest(project, logstore
, topic, source, logGroup);
            client.PutLogs(req2);
    }
    }
}
```